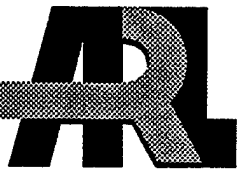
ARMY RESEARCH LABORATORY



Predicting Fragmentation Propagation Probabilities for Ammunition Stacks

John Starkenberg
Kelly J. Benjamin
Robert B. Frey

ARL-TR-949 January 1996

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1996	Final, February - December 1994

4. TITLE AND SUBTITLE

Predicting Fragmentation Propagation Probabilities for Ammunition Stacks

5. FUNDING NUMBERS

4G031-412-T5

6. AUTHOR(S)

John Starkenberg, Kelly J. Benjamin, and Robert B. Frey

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-WT-TB
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER

ARL-TR-949

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

By combining several existing models, we have developed a tool for estimating the probabilities associated with the propagation of detonation or burning between ammunition stacks. The models include the FRAGHAZ program for the Monte Carlo treatment of fragment trajectories and the accumulation of hit probabilities, the Jacobs-Roslund criterion for initiation of detonation and the ballistic limit condition for initiation of burning. We have applied this tool to artillery ammunition and missile stacks. Since the appropriate fragmentation input data was not always available, notably in the case of missiles, we developed methods of estimating this data. Single artillery round donors were shown to require a near-direct hit in order to initiate detonation in either artillery ammunition or missile acceptor stacks. Artillery ammunition donor stacks were shown to be much more lethal than missile donor stacks, and missile acceptor stacks were shown to be more vulnerable to the propagation of burning than artillery ammunition acceptor stacks.

14. SUBJECT TERMS

fragments, ammunition fragments, en masse detonation, quantity-distance

15. NUMBER OF PAGES

79

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

LIST OF FIGURES

INTENTIONALLY LEFT BLANK.

LIST OF TABLES

INTENTIONALLY LEFT BLANK.

1. BACKGROUND

In conjunction with a study of the benefits of insensitive munition (IM) technology, a need arose to develop a methodology for prediction of the probabilities of propagation of reaction (detonation or burning) and mechanical damage between a detonating stack of ammunition and its neighbors as functions of the distance between them. Such a capability could provide input to further analyses to predict losses in a variety of combat scenarios.

An informal survey of propagation accident reports (summarized in Appendix A) indicates that the most common mechanism of reaction propagation involves ignition of fires (often in combustible packaging) by fragments, debris, or firebrands from the source explosion and subsequent violent reaction of munitions in those fires. The resulting chain of events may take hours or even days to unfold. This scenario appears to be too complex and variable to model at the present time.

In some other cases, fragments from the source (donor) explosion can promptly damage, ignite mild to violent burning in, or detonate the energetic components (not including any combustible packaging) of munitions in nearby (acceptor) stacks. When detonation results, reaction can propagate further by the same mechanism, rapidly consuming large quantities of ammunition. When only burning or mechanical damage results, the acceptor stack may be totally or partially destroyed but no further propagation by the fragment mechanism ensues. This scenario is amenable to modeling.

In order to span the range of munition vulnerability, we wanted to obtain predictions applicable to typical (thick-walled) artillery projectiles and (thin-walled) missiles stacked on pallets. We also wanted to obtain predictions for a single artillery projectile donor representing an attacking munition which might be required to start the donor-acceptor chain in an analysis. We have chosen palletized and single M107 155-mm projectiles and palletized TOW-2A missiles as representative items.

2. APPROACH

We determined that the following elements are required to successfully model propagation among stacks of these items:

- descriptions of the stack storage arrangements

- arena test data for donor stack munitions describing the initial fragment mass, velocity and shape distributions, or estimates of such data

- a treatment of fragment trajectories to determine hazard probabilities

- descriptions of the vulnerable components of the acceptor stack munitions

- criteria for the initiation of detonation and burning as well as for mechanical damage.

The pallet arrangements commonly used for the weapons of interest have been determined. These form the building blocks for larger stacks.

Arena test data is generally available only for warheads whose performance is measured by fragmentation statistics. That is, data is available for the M107 projectile but not for the shaped-charge warhead and rocket motors found in the TOW-2A. We had to develop a method to estimate the data for the latter items.

We decided to modify an existing computer program called FRAGHAZ (McClesky 1988) in order to compute fragment trajectories and the desired probabilities. FRAGHAZ was developed to predict the hazard to a human target due to fragmentation from an exploding ammunition stack. Fragmentation data for a number of munitions is provided with the program.

Descriptions of the vulnerable components of these weapons were sometimes difficult to come by. They were obtained from a variety of sources. In some cases, best guesses were used. In all cases, the geometries were simplified to simple cylindrical metal shells filled with energetic materials.

An initiation criterion for detonation is well known and a simple criterion for burning may be developed from research data pertinent to this phenomenon.

3. FRAGHAZ

FRAGHAZ uses a fourth order Runge-Kutta scheme to compute trajectories for each fragment specified in the arena test input data. The trajectories include effects of ricochet and wind (if desired).

All the fragments are assumed to emanate from a single vertical line extending from the base to the top of munitions in the face of the stack as shown in Figure 1. This defines the stack axis. The initial fragment height is selected at random within this range. The initial fragment velocity and elevation angle are selected at random in a range near the values given in the input data. Parameters used to determine the fragment drag coefficient are determined as functions of a randomly selected value. The program replicates these trajectory computations many times with randomly selected environmental conditions. While FRAGHAZ provides "Monte Carlo" and "Full Factorial" options, the random selections referred to in the foregoing description apply only to the Monte Carlo option.

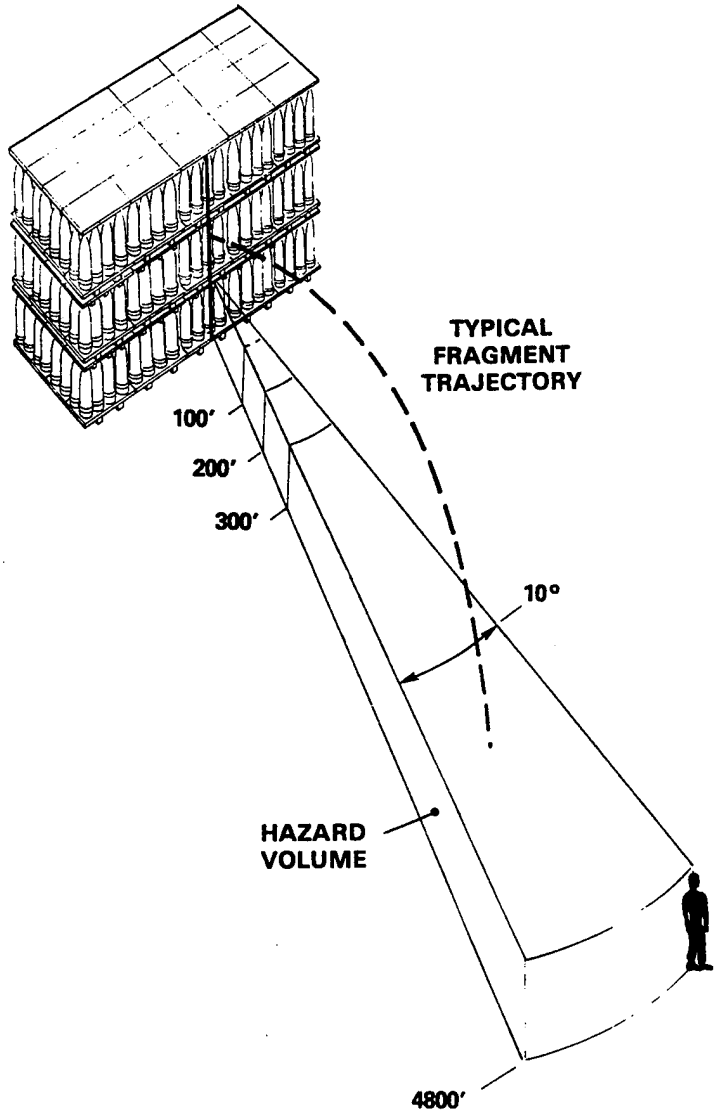


Figure 1. FRAGHAZ representation of a donor stack showing the stack axis and hazard volume (from NSWC TR 87-59).

A downrange "hazard volume," also shown in Figure 1, is defined. This consists of a cylindrical sector originating at the stack axis having a specified angular width (equivalent to the azimuthal sector associated with arena test data collection) and having the height of the target (nominally a 5.72-ft tall standing man). It is divided into a number of 100-ft annular segments. As each fragment passes through a hazard-volume segment, various hit probabilities and fragment densities are accumulated and represented as functions of the downrange distance associated with the midpoint of the segment. A determination of whether or not the fragment is hazardous based on its kinetic energy is made. The hazard hit probabilities depend on the ratio of the presented area of the target to the presented area of the hazard-volume segment with respect to hazardous fragments as shown in Figure 2.

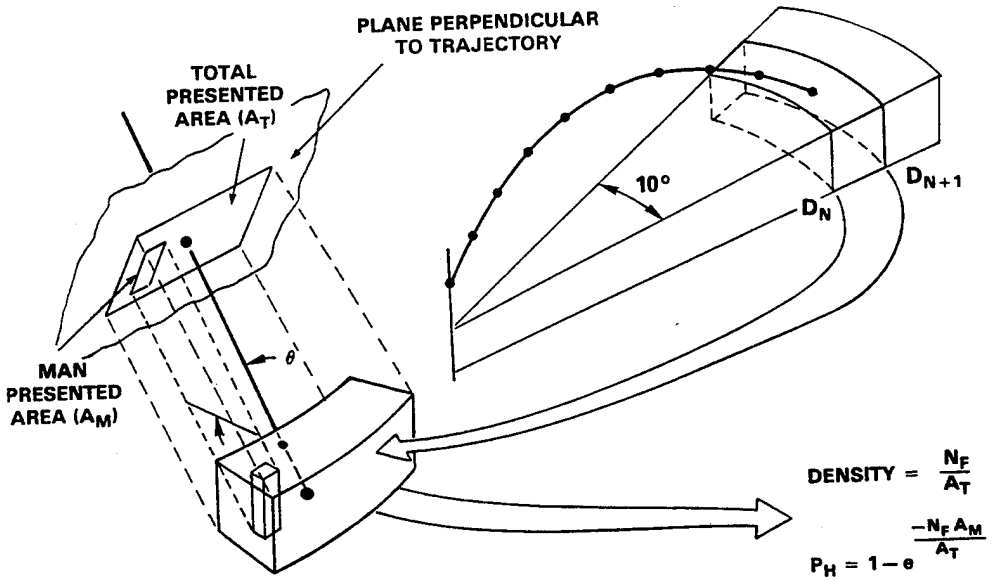


Figure 2. FRAGHAZ representation of target (acceptor) vulnerability and algorithm for determining hazard probability (from NSWC TR 87-59).

4. FRAGPROP

While its basic features were retained, the modifications to FRAGHAZ were extensive. They proceeded in two phases: the first to streamline the existing code and the second to implement the required new models. The resulting code was renamed FRAGPROP. A listing of the source code is provided in Appendix B. A flowchart applicable to both FRAGHAZ and FRAGPROP is shown in Figure 3.

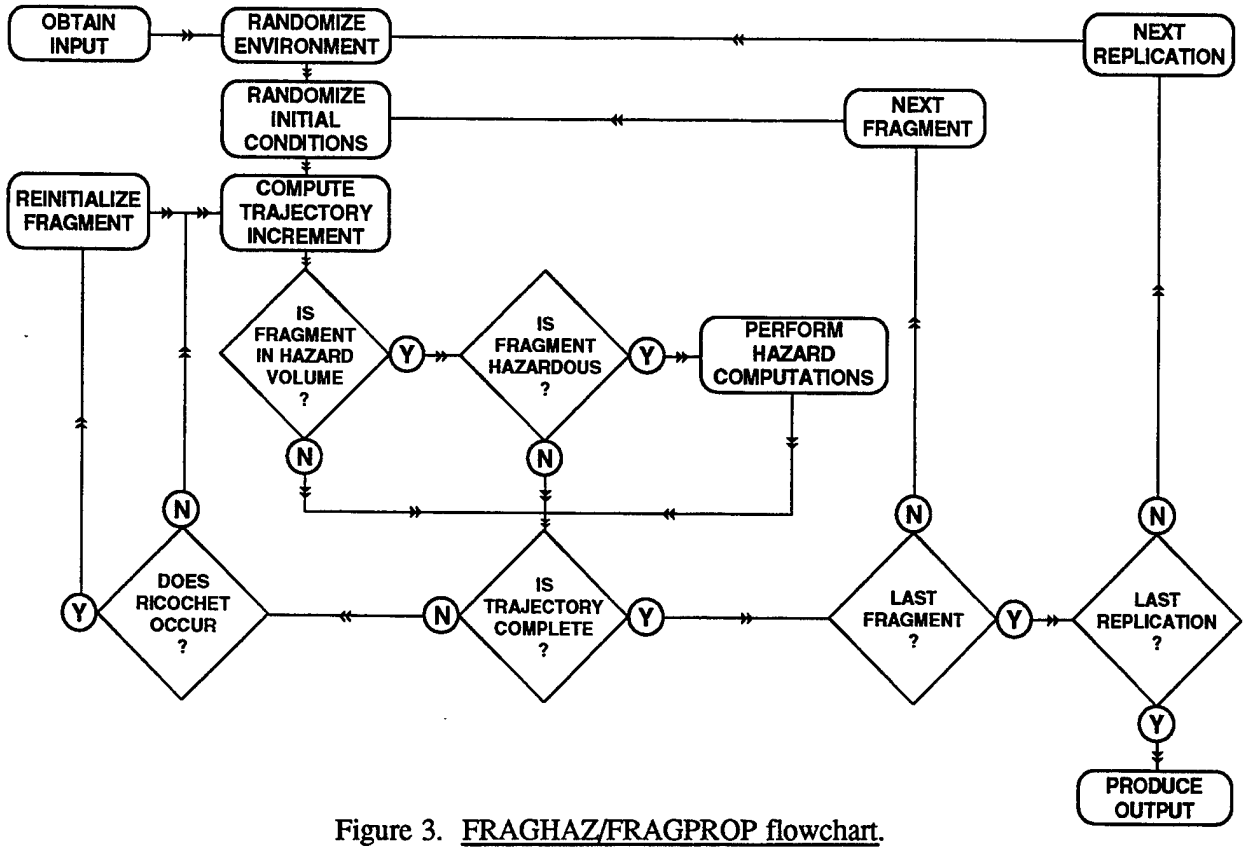


Figure 3. FRAGHAZ/FRAGPROP flowchart.

The original program made minimal use of modularization and was unwieldy to work with. We broke the code down into subroutines. As a result, the overall logic is easily followed in the main program. We changed the variable names to more descriptive forms which conform to FORTRAN naming conventions. In conjunction with this, we eliminated most of the type declarations. Many of the variables and computations were unnecessary for our purposes and we eliminated them. Some expressions, although correct, were recast in order to eliminate mixed-type operations. Elseif constructions were eliminated for clarity and less convoluted logic was substituted where possible. Execution speed was increased by moving expressions which had been repeatedly evaluated within loops to positions outside those loops and by eliminating redundant computations. For simplicity, we also eliminated the Full Factorial option.

New features and algorithms were also incorporated. Input was reorganized to read separate files for the run parameters, the donor description, and the acceptor description. Fragmentation data input was rewritten to reject fragments not meeting user-specified criteria for minimum mass and initial elevation angle. The downrange segment size (which had been fixed at 100 ft) was generalized to accept user specification. Computation of vulnerable areas for the acceptor elements and algorithms for determination of the detonation, burn, and mechanical damage lethality of each fragment including the effects of

container penetration were added. Finally, a determination and report of the minimum range at which the analysis is applicable was implemented. These algorithms are described in detail in the following sections.

5. DONOR MODELS

5.1 Stack Description. The donor stack is described as in the original FRAGHAZ code. The height of the bottom of the stack and the height of munitions in the stack are required to provide bounds for the initial height of each fragment. The size of the stack contributing to downrange fragmentation is described by specifying the number of "interaction areas" present on the downrange face of the stack. An interaction area is formed by each side-by-side pair of munitions. Thus, an arrangement of three projectiles in a row forms two interaction areas. If two such arrangements are stacked one on top of the other, four interaction areas result. However, if two such arrangements are set side by side (six rounds in a row), five interaction areas are produced. Interaction areas are considered instead of individual munitions because the effect of focusing along the plane of symmetry between the rounds augments the fragment velocities. Interaction areas are specified along with arena test data for multiple munitions. However, where data for single munitions (including estimated data) must be used, the donor unit is the single munition and the number of munitions on the downrange face specifies the stack size.

5.2 Arena Test Fragmentation Data. Arena test data forms the basis for describing the fragmentation characteristics of a particular munition. Data for interacting M107 projectiles including parameters for 215 fragments is provided with FRAGHAZ. Each fragment recovered in the test is characterized by five parameters: the polar zone of its origin, its mass, velocity, and area to mass ratio as well as the ratio of its maximum to average presented area (to account for the effects of tumbling on aerodynamic drag).

Fragment multipliers (McClesky 1988) are used to normalize the data for scaling to any stack size (i.e., any number of interaction areas or munitions) and any hazard volume azimuthal sector width. One fragment multiplier must be supplied for each polar zone represented in the data. The formula for computing the fragment multiplier for the *i*th polar zone, ϕ_i, is

$$M_F(\phi_i) = \frac{1}{N_I \psi_s(\phi_i)} ,$$

where N_I is the number of interaction areas (or single munitions, if appropriate) in the face of the stack *from which the data was collected* and $\psi_S(\phi_i)$ is the azimuthal sector width over which fragments were collected in the ith polar zone.

5.3 Adapting Arena Test Fragmentation Data. Arena test data for single munitions tabulated in the Joint Munitions Effectiveness Manual of Fragmentation Data (FM 101-62-3) is not in the FRAGHAZ format and does not provide all of the parameters required. Data for individual fragments is not provided. Rather, the mean fragment weight and integrated number of fragments for 50-grain weight increments are tabulated along with the initial velocity for each polar zone. A shape factor, K (=0.5126 $cm^2/g^{2/3}$ for the M107), for all fragments, is given and can be used in the expression,

$$\frac{a_{f_{avg}}}{m_f} = \frac{K}{m_f^{1/3}},$$

to estimate the area to mass ratio of a fragment.

We expanded this data to generate a table of 870 individual fragments (as required by FRAGPROP) by rounding the integrated number of fragments for each polar zone and fragment mass increment to an integral value and generating table entries for that number of individual fragments. These fragments were each assigned the mean weight of the associated weight increment and the mean velocity of the associated polar zone. In doing this, it was possible to eliminate small fragments. Since the ratio of the fragments' maximum to average presented area is not given, we used the average value taken from the FRAGHAZ input data for M107 stacks.

$$\frac{a_{f_{max}}}{a_{f_{avg}}} = 1.479.$$

For this arena test, there is only one unit in the stack and the integrated number of fragments is adjusted to apply to the entire 360° azimuthal sector. Thus, the fragment multipliers are independent of polar zone ($M_F = 1/360 = 2.7778 \times 10^{-3}$).

A capability to invert the polar distribution of the donor fragmentation data was added to FRAGPROP in order to allow representation of a nose-down attacking projectile as the initial event in a propagation chain.

5.4 Estimating Fragmentation Data. The TOW-2A missile is not a fragmenting weapon and fragmentation data is not available. We needed to estimate data having a representative distribution of values of the five parameters required by the FRAGPROP input. Several analytical techniques applicable to single cylindrical munitions are available for this purpose. The fragmentation unit in this case is a single munition rather than an interaction area. Thus, focusing effects are ignored and the number of rounds is specified in lieu of the number of interaction areas. Geometric data required to represent the TOW-2A warhead and rocket motor as cylindrical components and energetic material performance data as needed for use with the following analyses are summarized in Appendix C.

The Mott equation (see Victor 1994) describes the distribution of fragment mass. The fraction, f, of fragments having a mass greater than m_f is given by

$$f(m_f) = \sqrt{\exp(-2m_f/\overline{m_f})} \ ,$$

where $\overline{m_f}$ is the average fragment mass. This equation may be solved for m_f.

$$m_f(f) = -\frac{1}{2}\overline{m_f}\ln(f^2) \ .$$

By selecting random values of f which are evenly distributed between 0 and 1, this equation may be used to generate sets of fragment masses having the Mott distribution. The total number of fragments is given by

$$n_f = \frac{m_c}{\overline{m_f}} \ ,$$

where m_c, the total mass of all fragments, may be identified with the casing mass. Victor also gives the average fragment mass as

$$\overline{m_f} = \frac{A}{p_d}\sqrt{1+\mu/2}\left[\frac{h_c}{d_x}\left(d_x+h_c\right)^{3/2}\right],$$

where A is a constant, p_d is the detonation pressure, μ is the ratio of the mass of the casing to the mass of the charge, h_c is the thickness of the casing, and d_x is the diameter of the explosive or propellant charge (equal to the inside diameter of the casing). A = 676.2 g-kbar/in$^{3/2}$ for mass in g, pressure in kbar and distance in inches. The Kamlet-Jacobs formula may be used to estimate the detonation pressure (in GPa) if it is not otherwise known.

$$p_d = 1.558\rho_o^2 N\sqrt{MQ},$$

where ρ_o is the unreacted density in g/cm^3, N is the number of moles of detonation product per gram of unreacted material, M is the average molecular weight of the detonation product gases, and Q is the chemical energy of the detonation reaction in cal/g.

No method exists for predicting a distribution of fragment velocities. However, the Gurney equation (see Dehn 1984) may be used to estimate a single velocity for all fragments.

$$v_f = \frac{\sqrt{2E}}{\sqrt{\mu+1/2}},$$

where $\sqrt{2E}$ is a property of the energetic material (having the units of velocity) known as the Gurney constant. Its value may be found in the literature or estimated (in milliseconds) using the equation of Kamlet and Finger.

$$\sqrt{2E} = 233\frac{\sqrt{p_d}}{\rho_o^{0.6}},$$

for density in g/cm^3 and detonation pressure in kbar. Although this approach gives only one value of the fragment velocity, randomization over a narrow range takes place within FRAGPROP.

For estimating the area to mass ratio of a fragment, Victor (1994) gives the shape factor as $K = 0.5199\ cm^2/g^{2/3}$ for randomly shaped fragments. We are not aware of any method to estimate the ratio of a fragment's maximum to average presented area, so we assumed a fixed value of 1.5.

The TOW-2A storage orientation is horizontal in contrast to that of the M107. Thus, it is necessary to interpret the (rather narrow) polar distribution of fragments with respect to the munition axis as an azimuthal distribution with respect to the stack axis. This is assumed to produce an azimuthal sector width of only 20°. Because the missile components are treated as horizontal cylinders, the polar distribution with respect to the stack axis is assumed uniform. Since the azimuthal sector width is independent of polar zone, the fragment multiplier is a constant ($M_F = 1/20 = 0.05$).

We wrote a short program (incorporating the FRAGHAZ random number generator) to produce FRAGPROP input data for a general warhead and rocket motor combination in accordance with the foregoing algorithms and used it to generate data for 365 fragments from the TOW-2A warhead and flight motor. These components produce both aluminum and steel fragments. We neglected fragments from the launch motor, which has a small diameter. The program is listed in Appendix D.

6. ACCEPTOR MODELS

6.1 Stack Description. The basic acceptor stack description is given by providing dimensions of height (H_a), width (W_a), and depth (D_a). The storage orientation may be vertical, as with the M107, or horizontal, as with the TOW-2A. The vulnerable components of an acceptor may include a warhead, rocket motor, or both.

6.2 Vulnerable Areas. Vulnerable areas at the front and top of the stack are required. Side and back areas are excluded as the acceptor stack is assumed to face the donor. These depend on the weapon dimensions and the storage orientation (horizontal or vertical).

For the purpose of determining hit probabilities the total front and top areas of the stack are considered vulnerable. The front vulnerable area is

$$A_{fh} = H_a W_a \ ,$$

and the top vulnerable area is

$$A_{th} = D_a W_a \ .$$

For mechanical damage, the vulnerable areas are the areas presented to the front and top faces of the stack of the entire weapon (represented as a cylinder of diameter D_w and length L_w) multiplied by the number of weapons in that face of the stack (N_f or N_t). The vulnerable areas depend on the storage orientation. For vertical storage, the front vulnerable area is

$$A_{fm} = N_f D_w L_w \ ,$$

and the top vulnerable area is

$$A_{tm} = N_t \pi \left(D_w/2\right)^2 \ .$$

For horizontal storage, the front vulnerable area is unchanged.

$$A_{fm} = N_f D_w L_w \ ,$$

while the top vulnerable area is

$$A_{tm} = N_t D_w L_w \ .$$

Similarly, for detonation and burning, the *maximum* vulnerable areas are those of the energetic materials in the warhead and/or rocket motor (cylinders of diameter D_x and length L_x) presented to the front and top faces of the stack. For vertical storage, the front maximum vulnerable area is

$$A_{fx_{max}} = N_f D_x L_x \ ,$$

and the top maximum vulnerable area is

$$A_{tx_{max}} = N_t \pi \left(D_x/2\right)^2 \ .$$

For horizontal storage, the front maximum vulnerable area is

$$A_{fx_{max}} = N_f D_x L_x ,$$

and the top maximum vulnerable area is

$$A_{tx_{max}} = N_t D_x L_x .$$

Here, the subscript, x, represents either the warhead explosive or rocket motor propellant.

With respect to a particular fragment, the vulnerable areas for detonation and burning may be further limited depending on the velocity and diameter of that fragment. Below some critical threshold, none of the presented area of an energetic component is vulnerable. Above the threshold, part or all of the areas is vulnerable. Partial vulnerability is generally a function of the maximum obliquity, θ_{max}, from the normal to the surface of the weapon component at which an impacting fragment can produce reaction. Expressions for θ_{max} for detonation and burning are developed in the following sections.

6.3 Detonation Vulnerability. The vulnerability of a weapon component to initiation of detonation by fragment impact is described by the Jacobs-Roslund formula (Liddiard and Roslund 1993) for critical impact velocity. This formula applies to cylindrical projectiles having diameters greater than the failure diameter of the energetic material.

$$v_{jr} = \frac{a_{jr}\left(1+b_{jr}\right)\left(1+c_{jr}\frac{h_c}{d_f}\right)}{\cos\theta\sqrt{d_f}} ,$$

where a_{jr}, b_{jr}, and c_{jr} are characteristic constants which have been determined for a number of explosives. Values of the Jacobs-Roslund constants used in conjunction with our analysis are given in Appendix C. It is assumed that the fragment always strikes the vulnerable component with an average value of presented area. Thus, the fragment diameter, d_f, may be determined from the average fragment presented areas assuming a circular cross-section,

$$d_f = 2\sqrt{\frac{a_{f_{avg}}}{\pi}} .$$

The angle, θ, is the obliquity of the fragment with respect to the normal to the surface of the casing. The obliquity dependence of the critical velocity may be expressed as

$$v_{jr}(\theta) = \frac{v_{jr}(0)}{\cos\theta} .$$

We define the maximum obliquity that can produce detonation for a fragment with velocity, v_f, such that

$$v_{jr}(\theta_{max}) = v_f .$$

Thus,

$$\cos\theta_{max} = \frac{v_{jr}(0)}{v_f} ,$$

and θ_{max} is defined for $v_f \geq v_{jr}(0)$. This represents the critical condition. This model has been calibrated for steel fragments. However, we also used this calibration for the aluminum fragments produced by the TOW-2A warhead casing. As a worst case, we assumed that the fragments were flat-ended cylinders ($b_{jr} = 0$ with an appropriate value for c_{jr}) and that the critical diameter of the energetic material was always smaller than the diameter of the fragment. Since actual fragments rarely exhibit this morphology, a more realistic simulation might result from a random selection of the critical velocity between limits for flat- and round-ended cylindrical fragments.

6.4 Burning Vulnerability. The vulnerability of a weapon component to initiation of burning may be associated with perforation of the casing by the fragment. This is particularly true in the case of thin-walled munitions (Gilman). Some critical residual velocity, v_{rc}, may be required to produce reaction. The THOR velocity equation relates the residual velocity, v_r, to the initial velocity, v_f, for a penetrating fragment.

$$v_r = v_f - 10^{a_v} \left(h_c a_{f_{avg}}\right)^{b_v} m_f^{c_v} v_f^{d_v} / (\cos\theta)^{e_v} .$$

Again, it is assumed that the fragment strikes the vulnerable component with an average value of presented area. With $v_r = v_{rc}$ at $\theta = \theta_{max}$,

$$\cos\theta_{max} = \left[\frac{10^{a_v} \left(h_c a_{f_{avs}}\right)^{b_v} m_f^{c_v} v_f^{d_v}}{v_f - v_{rc}} \right]^{1/e_v} ,$$

and θ_{max} is defined for

$$v_f - v_{rc} \geq 10^{a_v} \left(h_c a_{f_{avg}}\right)^{b_v} m_f^{c_v} v_f^{d_v} .$$

This expression represents the critical condition. Values of the THOR constants used in conjunction with our analysis are given in Appendix C. Since appropriate values of the critical residual velocity are not known, we used $v_{rc} = 0$ as a worst case. This model has also been calibrated for steel fragments but, again, we used the same calibration for the aluminum fragments produced by the TOW-2A warhead casing.

6.5 Comparison of Detonation and Burning Thresholds. It is tempting to presume that the less violent burning response is easier to produce than detonation. However, comparison of the critical velocities predicted by these models shows that this is not always the case. Critical velocities for detonation and burning of a typical fragment are plotted as functions of casing thickness in Figure 4. For thin, easily perforated casings (e.g., ~1 mm for the TOW-2A), burning is produced by fragments having much lower velocities than those required to produce detonation. (This result would still hold if a nonzero critical residual velocity of expected magnitude were used.) For thicker casings (e.g., ~15 mm for the M107), casing perforation is extremely difficult to achieve but detonations may be produced at lower velocities due to the transmitted shock.

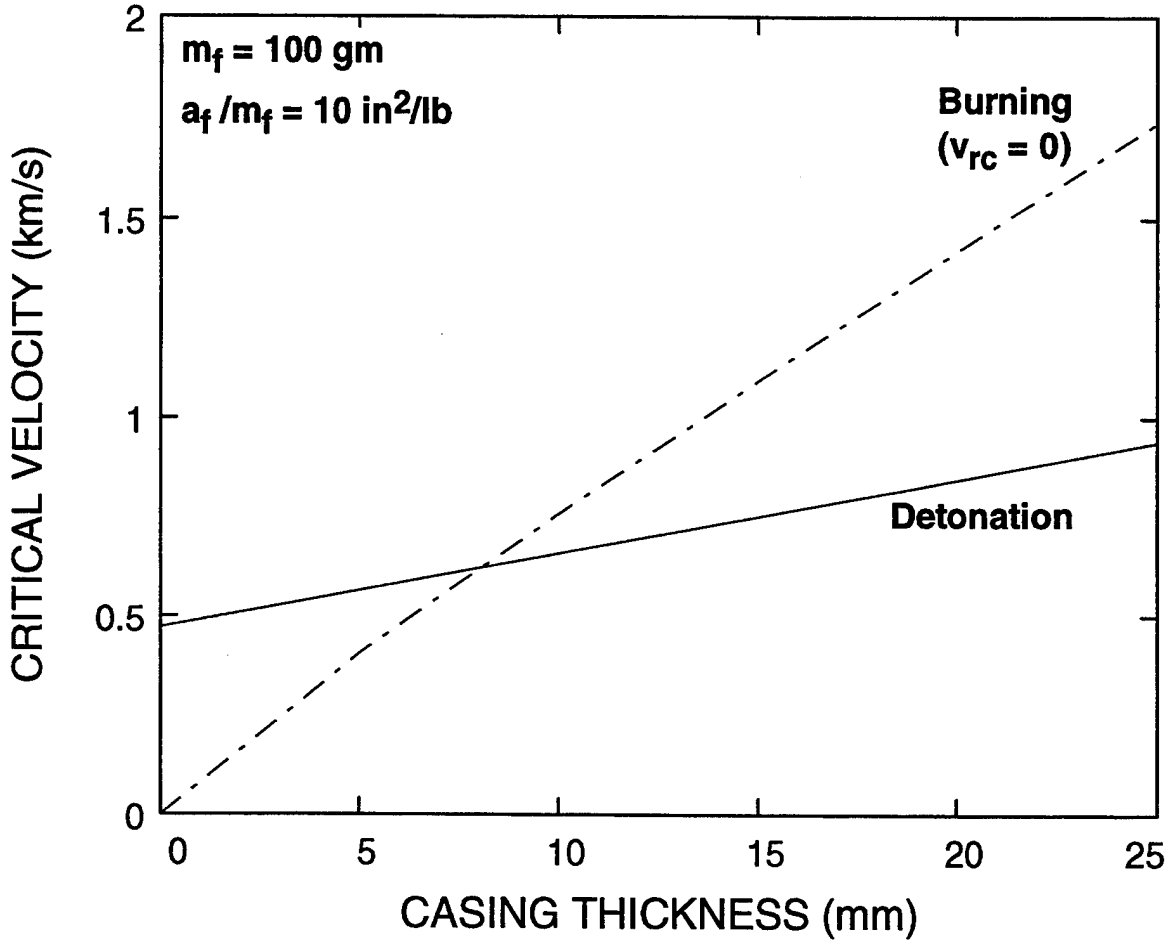


Figure 4. Comparison of critical velocities for detonation and burning.

The point of intersection of the two threshold curves varies with the shape of the fragment (represented by the area to mass ratio). For ratios lower than the 10 in^2/lb value used in Figure 4 the point of intersection is shifted toward high values of casing thickness. Thus, results are sensitive to assumptions regarding fragment orientation (presented area) at impact.

6.6 Vulnerable Area Reduction. If the angle subtended at the center of the cylinder representing a vulnerable component by the region on the surface over which an impacting fragment is lethal with respect to detonation or burning ($\theta \leq \theta_{max}$) is $2\beta_{max}$ as shown in Figure 5, then the reduction in the maximum vulnerable area of that component is given by

$$A = \sin\beta_{max} A_{max} .$$

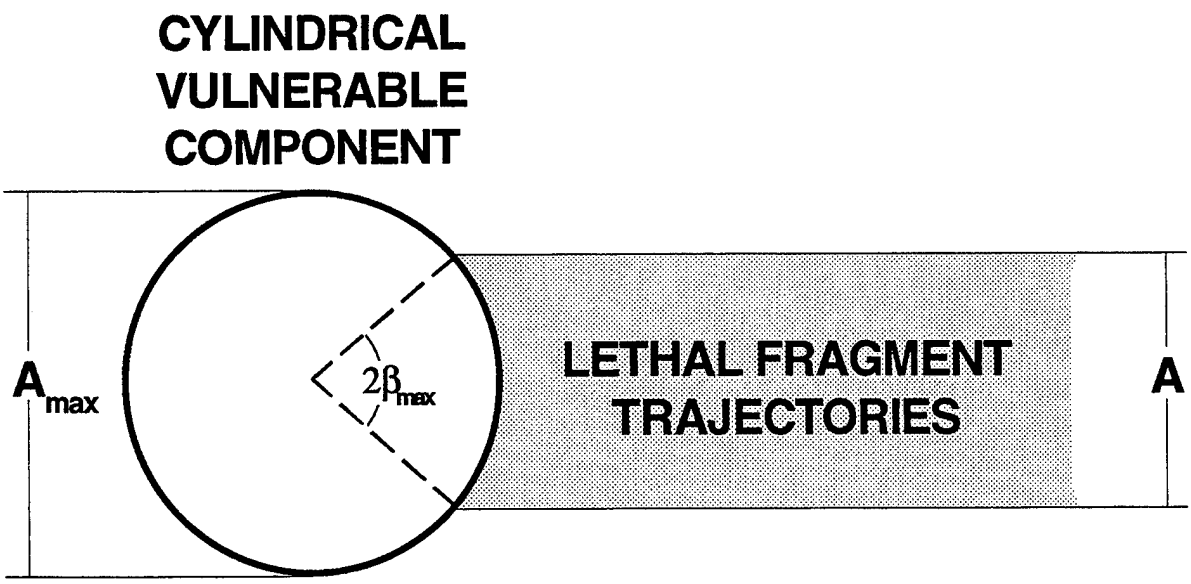


Figure 5. Angular region of vulnerability on a cylindrical charge.

The relationship between θ_{max} and β_{max} depends on the storage orientation. For horizontal storage,

$$\beta_{max} = \theta_{max}$$

as shown in Figure 6. For vertical storage, the elevation angle of the fragment trajectory, α_f, must be accounted for, as shown in Figure 7. Thus,

$$\cos\beta_{max} = \frac{\cos\theta_{max}}{\cos\alpha_f} .$$

6.7 Mechanical Damage Vulnerability. A single mechanical damage criterion based on kinetic energy applies to all weapon components. The stack is considered vulnerable to damage when the kinetic energy of a fragment exceeds a specified value, regardless of the obliquity. Accurate threshold values are not known, so we arbitrarily chose 400 and 50 ft-lb as the critical kinetic energy for the M107 and TOW-2A, respectively. The smaller value for the TOW-2A reflects its much thinner skin.

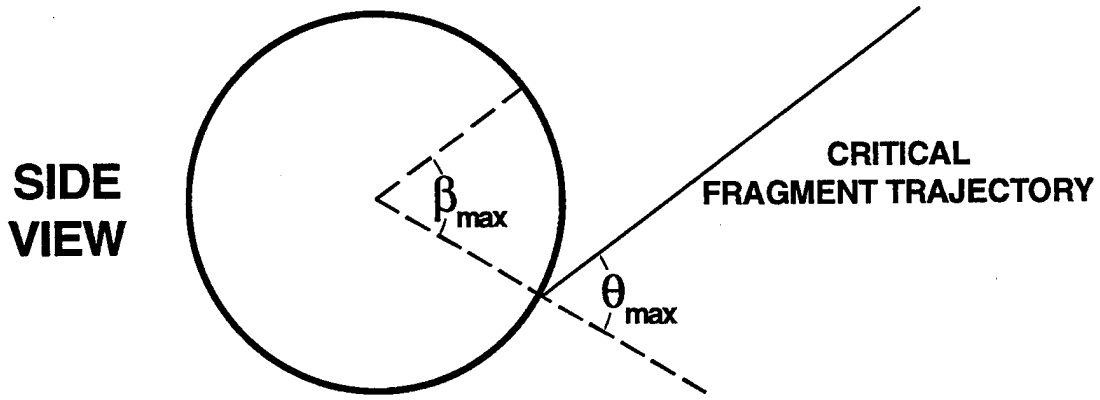


Figure 6. Relationship between the angular region of vulnerability and the maximum obliquity for horizontal storage.

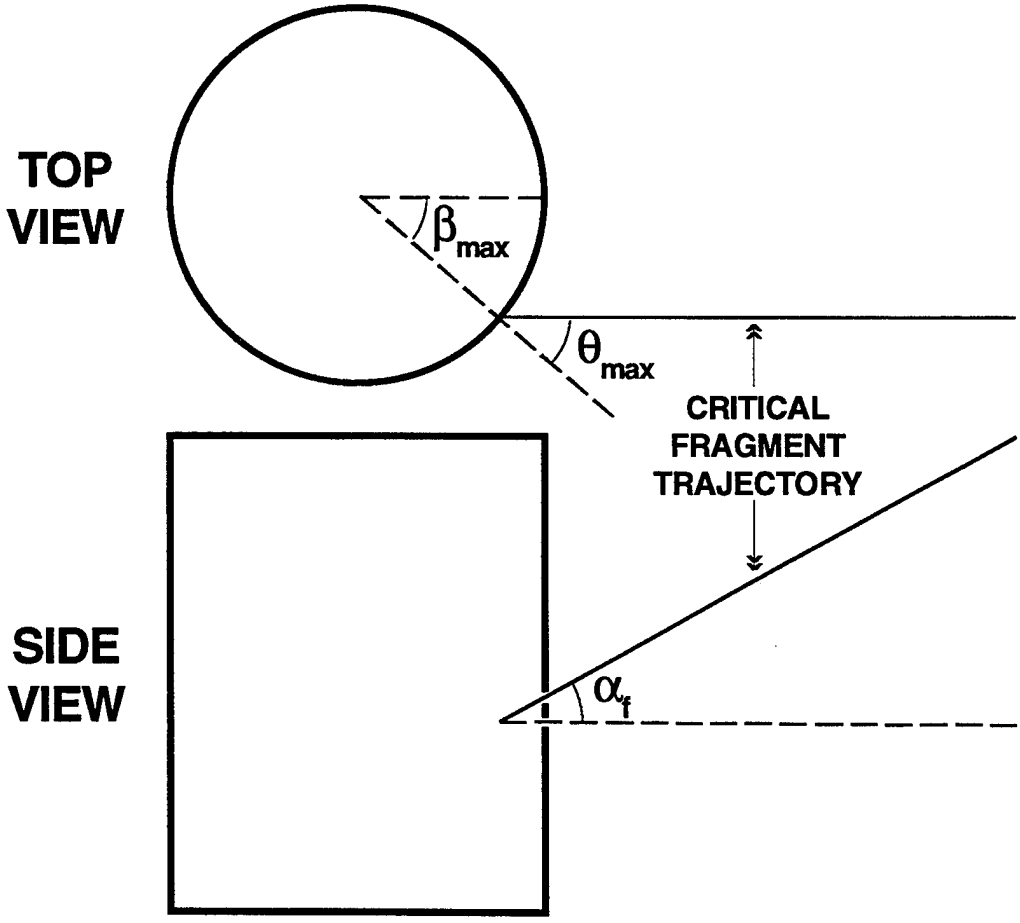


Figure 7. Relationship between the angular region of vulnerability, the maximum obliquity, and the fragment elevation angle for vertical storage.

7. CONTAINER PENETRATION

The TOW-2A is packaged in a steel container, and provisions were made in FRAGPROP for estimating the reduction in fragment mass and velocity associated with penetration of the container. This occurs both when fragments are launched from a donor stack and when they impact an acceptor stack. In addition to the THOR velocity equation, the THOR mass equation is required.

$$m_r = m_f - 10^{a_m}\left(h_c a_{f_{avg}}\right)^{b_m} m_f^{c_m} v_f^{d_m}/(\cos\theta)^{e_m} .$$

Use of the THOR equations with $\theta = 0$ yields the maximum residual velocity and mass, representing the worst-case scenario. The residual diameter is determined from the residual mass assuming that the area to mass ratio remains unchanged.

8. MINIMUM RANGE

The FRAGHAZ analysis is applicable as long as the angle subtended by the acceptor stack does not exceed the width, ψ, of the azimuthal sector associated with the donor fragmentation data. The minimum range is thus given by

$$r_{min} = \frac{W_a}{2\tan(\psi/2)} .$$

Note that r_{min} is zero for sectors wider than 180°.

9. COMPUTATIONAL CONFIGURATIONS AND RESULTS

9.1 Theater of Operations Considerations. Storage regulations applicable to basic load ammunition holding areas in theaters of operations limit the explosive quantity in any stack to 4,000 kg (8,818 lb). Such stacks must be separated by at least 77 m (253 ft). (See Army Regulation 385-64.) We considered M107 and TOW-2A stacks containing approximately this maximum weight.

Each M107 projectile contains 15.4 lb of Composition B. Thus, a pallet of eight projectiles contains 123.2 lb and a stack of 72 pallets contains 8,870.4 lb.

A TOW-2A missile contains 6.7 lb of LX-14 plus 0.2 lb of other high explosives in the warhead, 1.2 lb of M7 propellant in the launch motor and 7.0 lb of GCV propellant in the flight motor for a total of 15.1 lb. A pallet of 12 missiles contains 181.2 lb, and a stack of 48 pallets contains 8,697.6 lb.

9.2 Stack Arrangements. The arrangement of pallets affects the lethality and vulnerability of a stack. Consideration of the least and most lethal arrangements as well as the least and most vulnerable arrangements provides scope to a determination of propagation probabilities. Pertinent stack parameters may be estimated from pallet dimensions given in Appendix C.

Each M107 pallet contains eight vertical projectiles in an arrangement that is one projectile high, four projectiles wide, and two projectiles deep ($1 \times 4 \times 2$). The 72 pallets may be arranged in any permutation of $3 \times 4 \times 6$, where the first dimension is the height of the stack in pallets, the second dimension is the width of the stack in pallets, and the third dimension is the depth of the stack in pallets. Individual pallets retain their $1 \times 4 \times 2$ orientations for all permutations. Table 1 gives the number of interaction areas, stack dimensions, and total front and top vulnerable area associated with each permutation.

The $4 \times 3 \times 6$ pallet arrangement is both least lethal (having the fewest interaction areas) and least vulnerable (having the smallest vulnerable area, while the $4 \times 6 \times 3$ arrangement is both most lethal and most vulnerable.

Each TOW-2A pallet contains 12 horizontal missiles in an arrangement that is 3 missiles high, 1 missile wide, and 4 missiles deep ($3 \times 1 \times 4$). The 48 pallets may be arranged in any permutation of $3 \times 4 \times 4$. Table 2 gives the number of missiles on the front face, stack dimensions, and total front and top vulnerable areas associated with each permutation.

Based on the number of missiles on the front face, the $4 \times 4 \times 3$ pallet arrangement is most lethal. The other two arrangements are nominally equal in lethality. The $3 \times 4 \times 4$ arrangement is most vulnerable, while the $4 \times 3 \times 4$ arrangement is least vulnerable. The missile stacks are considerably larger and more vulnerable to being hit than the artillery projectile stacks.

Table 1. Lethality and Vulnerability Parameters for M107 Stacks

Pallet Arrangement ($n_h \times n_w \times n_d$)	Number of Interaction Areas on Face of Stack	Stack Dimensions			Vulnerable Areas (front + top) (ft^2)
		height (ft)	width (ft)	depth (ft)	
3 × 4 × 6	45	8.00	9.63	7.20	146.3
3 × 6 × 4	69	8.00	14.50	4.77	185.2
4 × 3 × 6	44	10.67	7.20	7.20	128.6
4 × 6 × 3	92	10.67	14.50	3.55	206.2
6 × 3 × 4	66	16.00	7.20	4.77	149.5
6 × 4 × 3	90	16.00	9.63	3.55	188.3

Table 2. Lethality and Vulnerability Parameters for TOW-2A Stacks

Pallet Arrangement ($n_h \times n_w \times n_d$)	Number of Missiles on Face of Stack	Stack Dimensions			Vulnerable Areas (front + top) (ft^2)
		height (ft)	width (ft)	depth (ft)	
3 × 4 × 4	36	10.06	20.25	14.64	500.0
4 × 3 × 4	36	13.41	15.17	14.64	425.3
4 × 4 × 3	48	13.41	20.25	10.96	493.4

The single M107 projectile donor may be erect or inverted and may detonate at any specified height above the ground. The arrangements producing the least and greatest lethality are not apparent. However, preliminary computations indicated that the inverted round at the surface may be considered least lethal and the erect round at a 50-ft elevation most lethal.

9.3 Propagation Probabilities and Distances. The graphics program developed in conjunction with FRAGPROP plots the probabilities of exceeding the criteria for propagation of detonation (labeled D), burning (B), and mechanical damage (M), as well as the overall (lethal and nonlethal) hit probability (H) on a logarithmic scale as a function of range. A typical example is shown in Figure 8. The dashed vertical line near the probability axis indicates the minimum range for which the analysis is accurate.

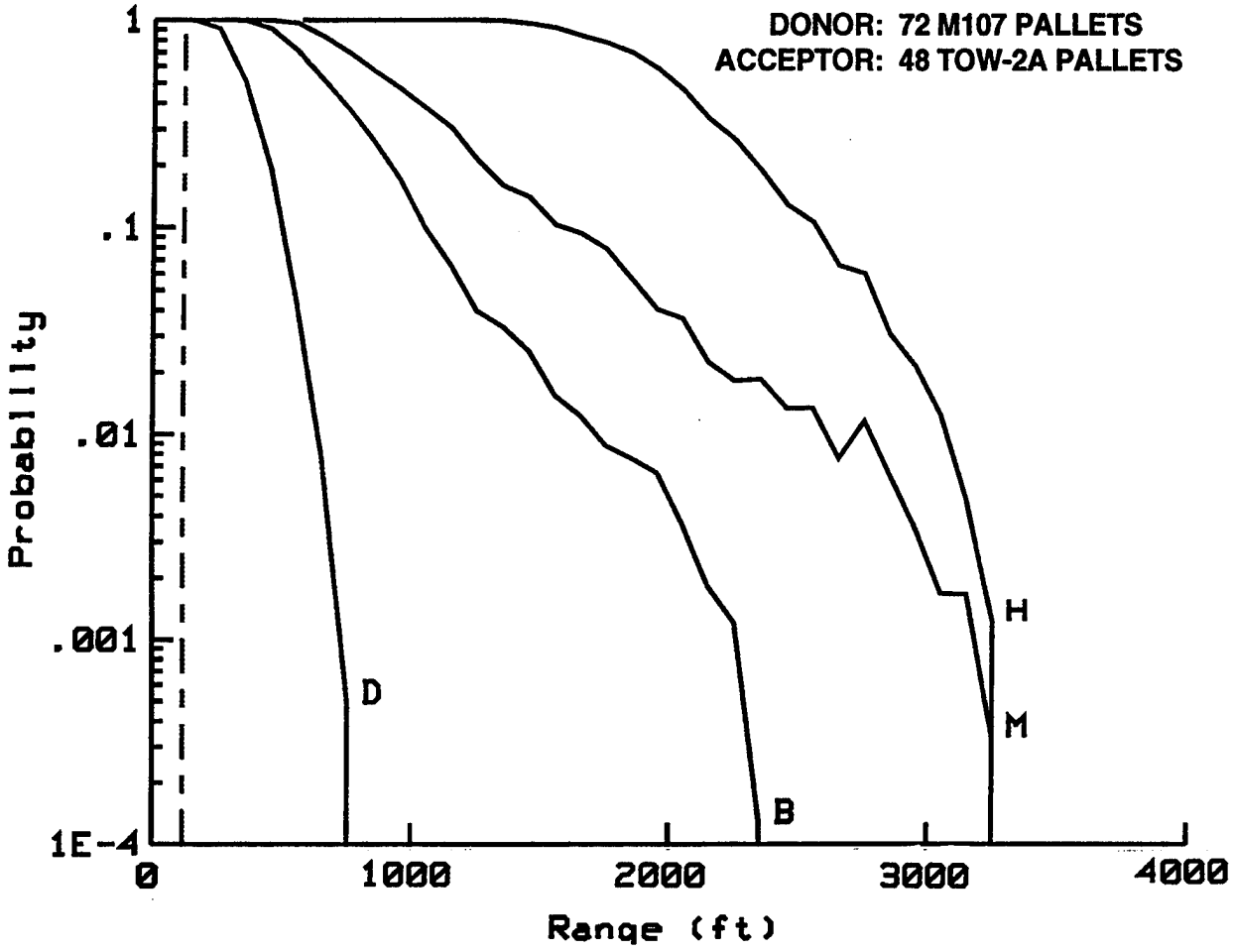


Figure 8. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for an M107 donor stack against a TOW-2A acceptor stack.

Generally, the probabilities decrease with increasing range and, theoretically, they can be expected to vanish entirely beyond some maximum range. For comparison purposes, it is often useful to consider the range at which a probability drops to some small value (e.g., 1%). The small probabilities predicted at long range may become nonmonotonic prior to vanishing. The ability to predict nonzero probabilities at long range is limited by the number of fragments in the donor data and the number of replications. Generally, the results are not accurate near vanishing probability. The number of replications required to produce monotonically decreasing probabilities varies inversely with the number of fragments describing the donor.

We made FRAGPROP computations using zero-wind conditions and soil constants varying between 0.5 and 4.0 for each of the three donors (single M107 projectiles, stacked M107 pallets, and stacked TOW-2A pallets) vs. both of the acceptors (stacked M107 and TOW-2A pallets). Preliminary computations were made to determine the maximum range required in each problem. Input conditions for the final computations are summarized in Table 3. These values depend on the donor and apply to both acceptors.

Table 3. FRAGPROP Computation Input Conditions

Donor	Number of Fragments	Number of Replications	Segment Size (ft)	Maximum Range (ft)
Single M107	870	50	50	2,400
M107 Stack	215	200	100	3,600
TOW-2A Stack	365	125	20	720

In order to determine the scope of probability values, two computations were made for each donor-acceptor pair: one for the least lethal donor stack arrangement against the least vulnerable acceptor stack arrangement and one for the most lethal donor stack arrangement against the most vulnerable acceptor stack arrangement. The graphics capability allows the four probabilities to be plotted as functions of range for two problems at a time as shown in Figures 9–14. Associated pairs of probability curves are joined by shading lines.

Results for single M107 donors against palletized M107 acceptors are shown in Figure 9. The probabilities decrease gradually with range. The distance at which the detonation propagation probability drops to 1% (on the upper curve of the pair) is 125 ft, while the 1% distance for burning is only 113 ft. Thus, burning propagation appears unlikely in this configuration. The probability of mechanical damage remains above 1% to a range of 687 ft, while the hit probability drops to 1% at 1,369 ft.

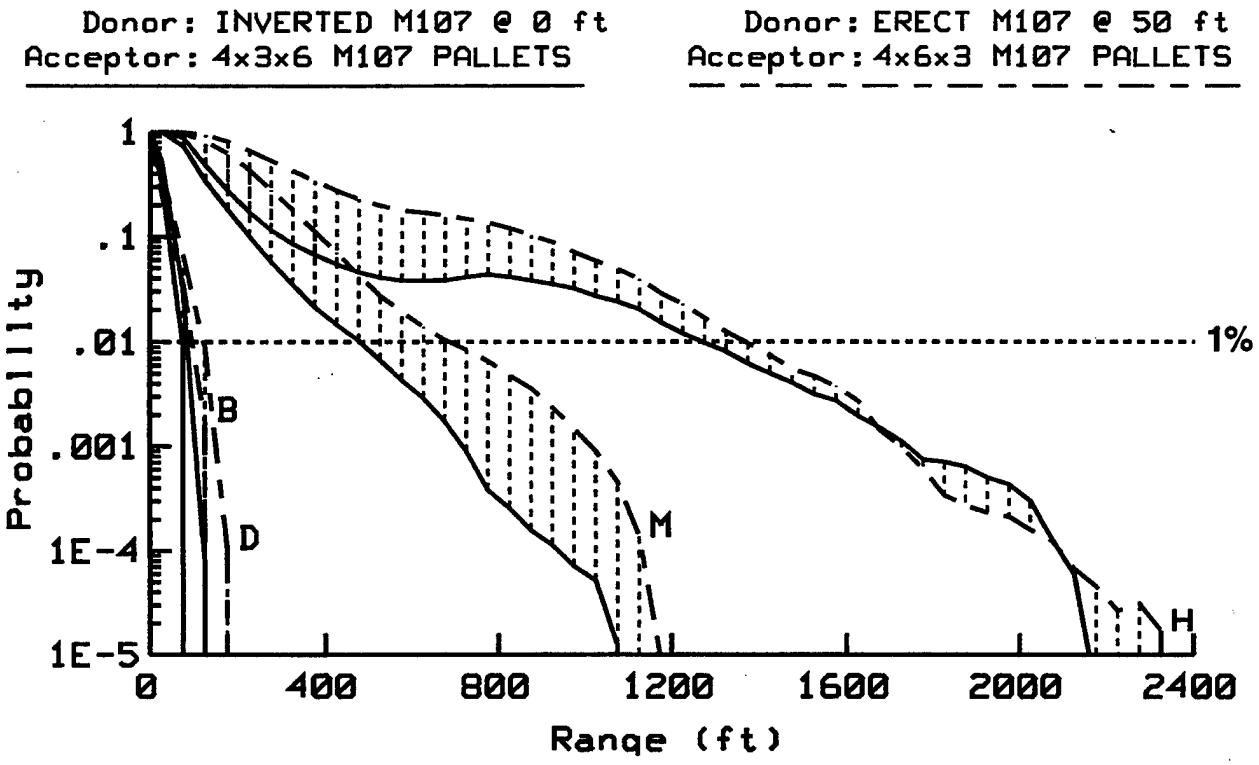


Figure 9. Probabilities of burning, mechanical damage, and hit as functions of range for single M107 donor projectiles against M107 acceptor stacks.

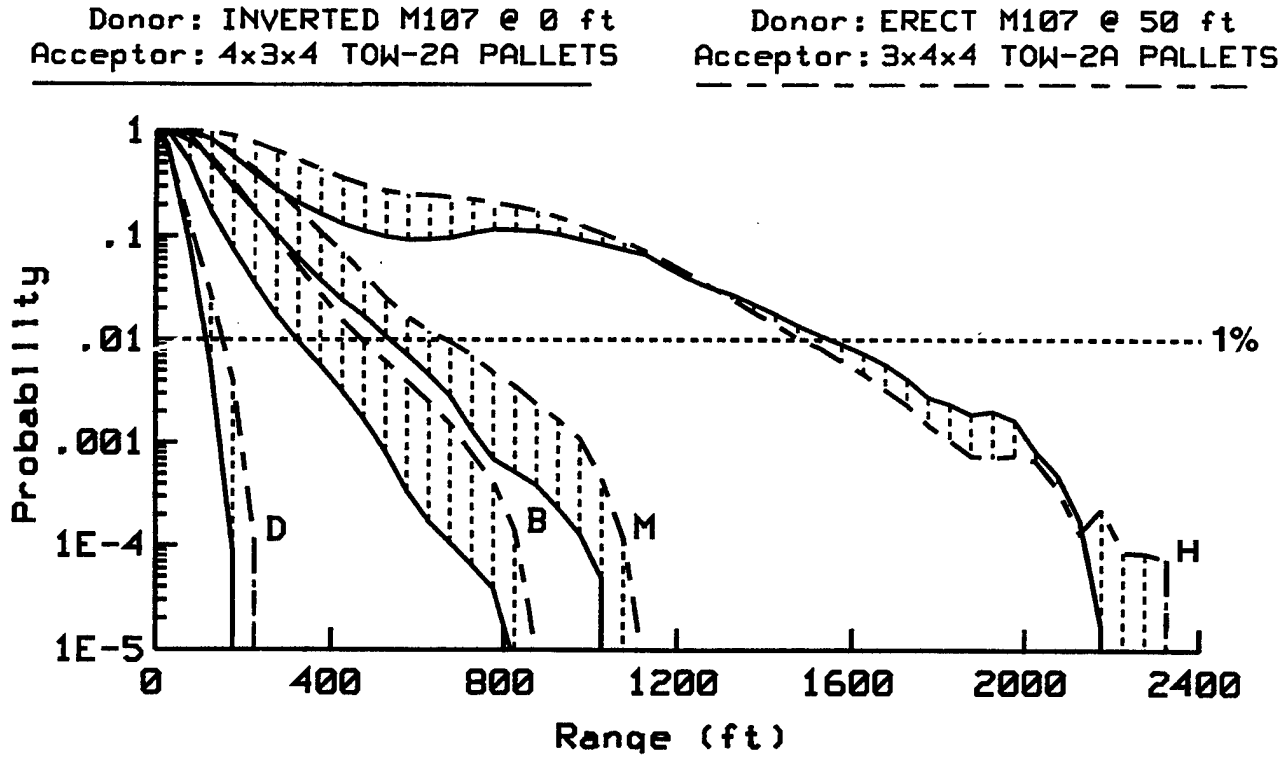


Figure 10. Probabilities of burning, mechanical damage, and hit as functions of range for single M107 donor projectiles against TOW-2A acceptor stacks.

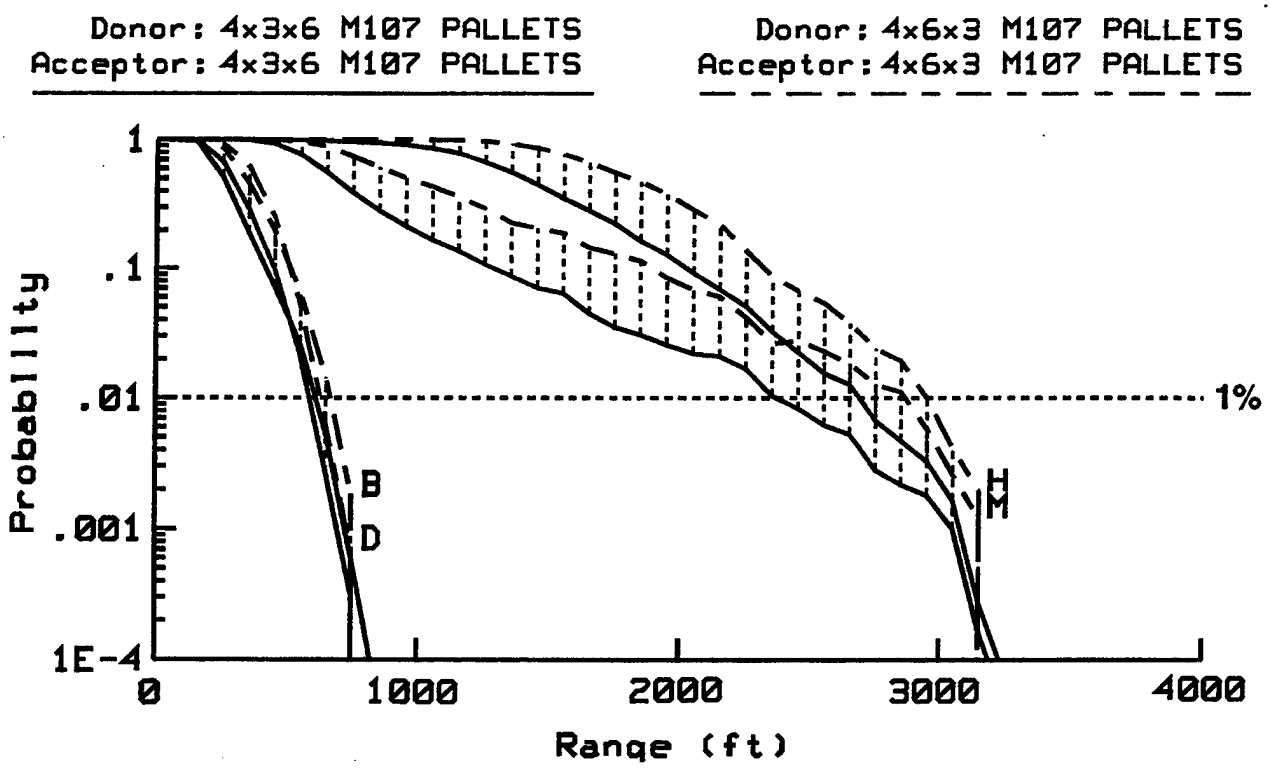


Figure 11. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for M107 donor stacks against M107 acceptor stacks.

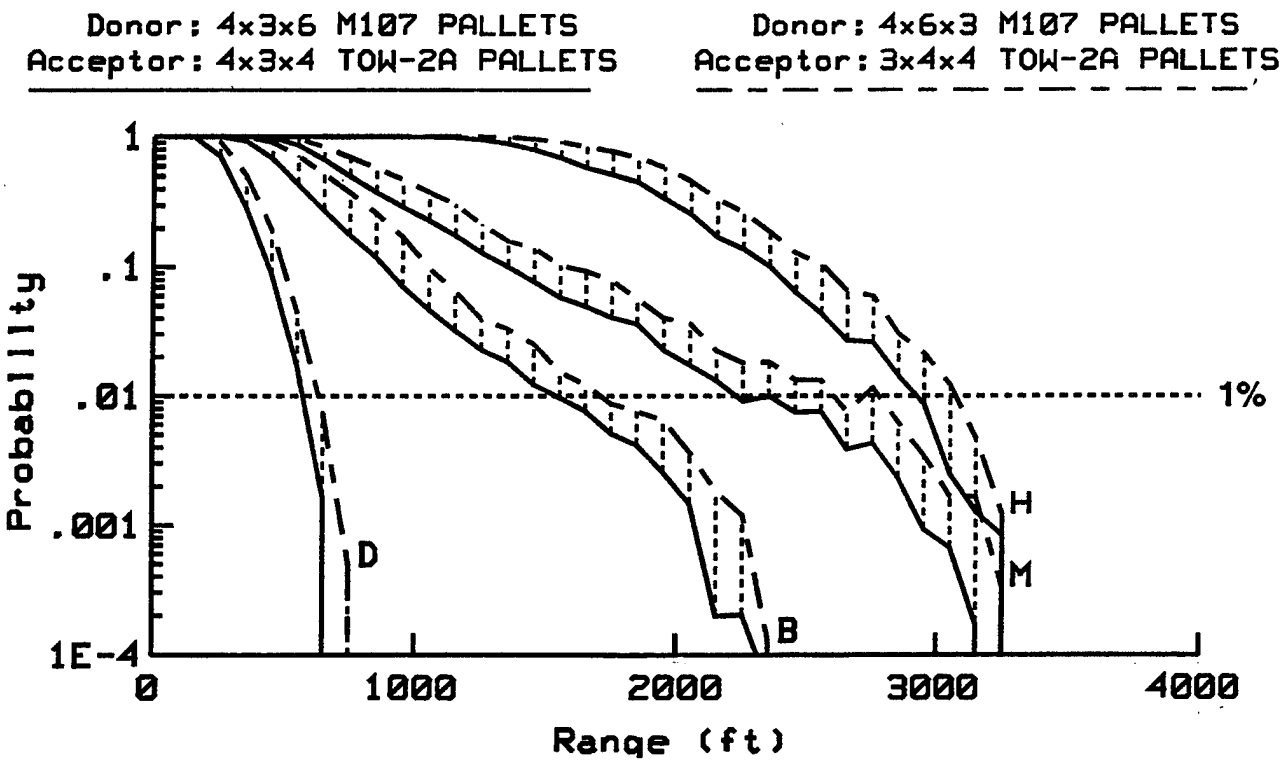


Figure 12. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for M107 donor stacks against TOW-2A acceptor stacks.

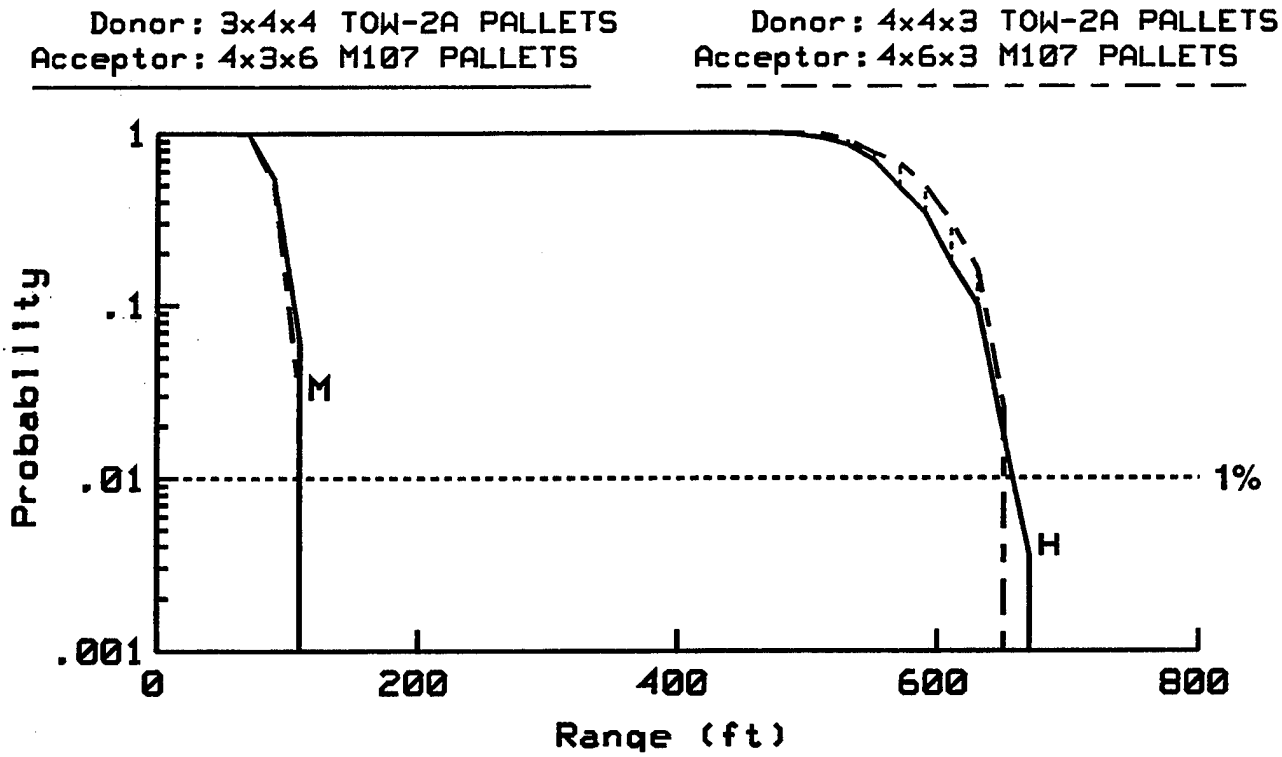


Figure 13. Probabilities of mechanical damage and hit as functions of range for TOW-2A donor stacks against M107 acceptor stacks.

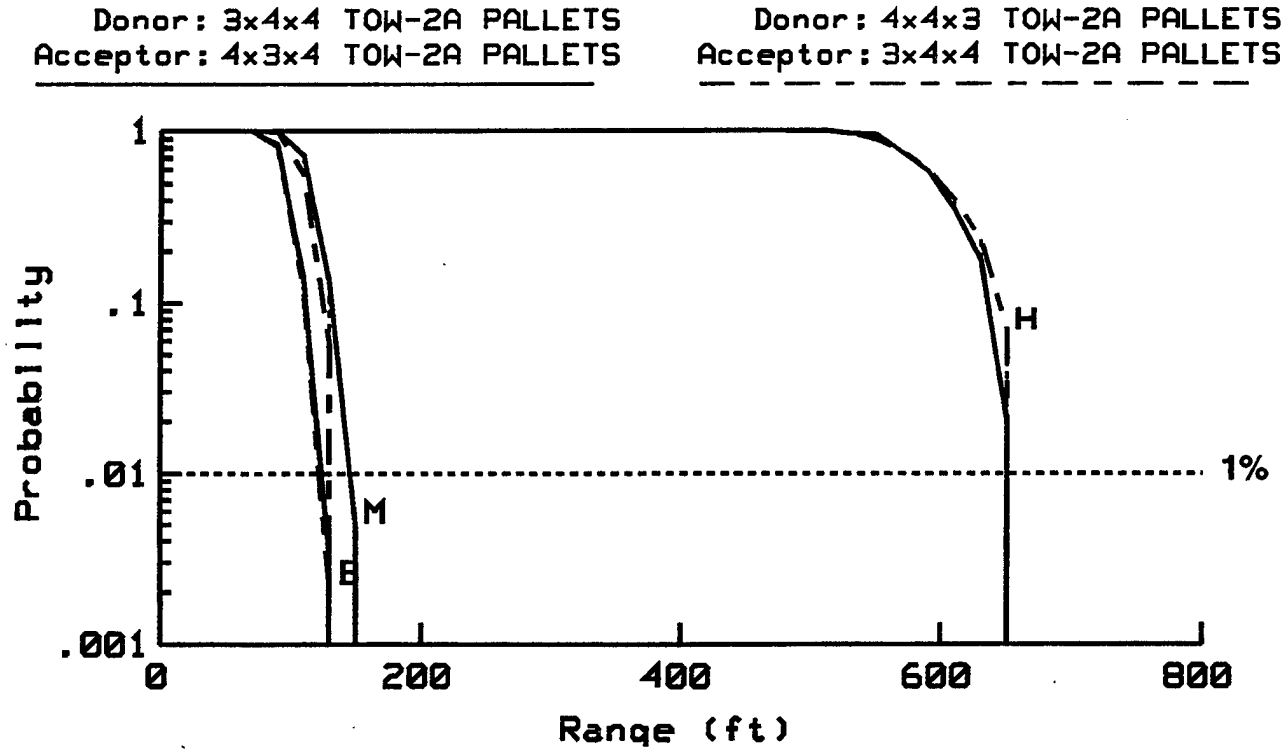


Figure 14. Probabilities of burning, mechanical damage, and hit as functions of range for TOW-2A donor stacks against TOW-2A acceptor stacks.

Similar results for single M107 donors against palletized TOW-2A acceptors are shown in Figure 10. The 1% distance for detonation is 162 ft, while a burn probability greater than 1% persists to a distance of 471 ft. The probability of mechanical damage drops to 1% at 666 ft. The range over which the overall hit probability remains above 1% is 1,551 ft.

Results for palletized M107 donors against palletized M107 acceptors are shown in Figure 11. Probabilities near unity persist to a greater range. This arrangement produces a 1% probability of detonation at 642 ft, while burning reactions propagate with a probability of 1% or greater over distances to 685 ft. The probability of mechanical damage remains above 1% to 2,871 ft. The 1% distance for the overall hit probability is 2,950 ft.

Similar results for palletized M107 donors against palletized TOW-2A acceptors are shown in Figure 12. This arrangement produces a 1% probability of detonation at 643 ft, while burning reactions propagate with a probability of 1% or greater over distances to 1,715 ft. A probability of mechanical damage greater than 1% persists to 2,779 ft. The 1% distance for the overall hit probability is 3,080 ft.

Results for palletized TOW-2A donors against palletized M107 acceptors are shown in Figure 13. Neither detonation nor burning propagation is predicted. The other probabilities remain high and then drop rapidly to zero with increasing range. Probabilities of mechanical damage exceeding 1% persist to 127 ft. The 1% distance for hit probability is 662 ft.

Similar results for palletized TOW-2A donors against palletized TOW-2A acceptors are shown in Figure 14. No detonation propagation is predicted, but burning propagation occurs at the 1% probability level out to a range of 129 ft. Probabilities of mechanical damage of 1% or greater persist to 149 ft. The 1% distance for hit probability is 667 ft. The fact that the burn and mechanical damage curves nearly coincide indicates that the arbitrary kinetic energy threshold for mechanical damage is too high.

The 1% propagation probability ranges discussed in the foregoing paragraphs are summarized in Table 4.

The lethality of the three donors varies substantially. Significant differences between the single and palletized M107 donors occur simply because of the number of munitions involved. The probabilities for

Table 4. 1% Propagation Probability Distances (ft)

Donor	Acceptor	Detonation	Burning	Mechanical Damage	Hit
Single M107	M107 Stack	124	113	687	1,369
	TOW-2A Stack	162	471	666	1,551
M107 Stack	M107 Stack	642	685	2,871	2,950
	TOW-2A Stack	643	1,715	2,779	3,080
TOW-2A Stack	M107 Stack	NP	NP	127	662
	TOW-2A Stack	NP	129	149	667

NP = no propagation

these donors drop slowly and the maximum ranges are considerably greater than the 1% ranges. TOW-2A donors are much less lethal due to the small fragments they produce. The character of the probability curves for these donors is different. Probabilities remain high until approaching the maximum range where they drop sharply.

The relative vulnerability of the two acceptors depends on the type of response. The TOW-2A is considerably more susceptible to burning than the M107. For the other responses, the 1% propagation distances are generally very similar. Although the (rather arbitrary) kinetic energy threshold is lower for TOW-2A acceptors, their probability of mechanical damage is reduced due to the presence of the container.

It is notable that this analysis (which includes a number of worst-case assumptions) does not preclude propagation of detonation from M107 stacks stored at the required 253-ft distance from either acceptor. At 250 ft, the probability of detonation propagation to an M107 stack is between 68% and 94% (depending on the stack configurations) and the probability of detonation propagation to a TOW-2A stack is between 71% and 91%.

10. SUMMARY AND CONCLUSIONS

By combining several existing models, we have developed a tool (FRAGPROP) for estimating the probabilities associated with the propagation of reaction between user-described ammunition stacks. The

acceptor responses considered include detonation of energetic materials, burning of energetic materials (but not of combustible packaging), and mechanical damage. The models include the FRAGHAZ program for the Monte Carlo treatment of fragment trajectories and the accumulation of hit probabilities, the Jacobs-Roslund criterion for initiation of detonation, and the ballistic limit condition for initiation of burning.

A number of difficulties arose in applying this tool to munitions of interest. Since the appropriate fragmentation input data was not always available, notably in the case of missiles, we developed methods of estimating this data. It was necessary to represent the vulnerability of explosives and propellants for which Jacobs-Roslund parameters are not available using estimates based on values for similar compositions. It was also necessary to assess the lethality of aluminum fragments with models intended for use with steel fragments.

Although only two weapons have been specifically analyzed, the responses of the thick-walled M107 and the thin-walled TOW-2A are expected to encompass the range of responses of a wide variety of munitions. The analysis was used to predict distances below which the propagation probabilities for each response exceed 1% (see Table 4). Based on these predictions, we conclude that the artillery ammunition donor stacks are much more lethal than the missile donor stacks (which cannot produce detonation) and the missile acceptor stacks are more vulnerable to the propagation of burning (but not of detonation) than the artillery ammunition acceptor stacks. However, different assumptions regarding the orientation of fragments at impact time might enhance sensitivity to burning.

11. REFERENCES

Dehn, J. T. "Models of Explosively Driven Metal." BRL-TR-2626, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, December 1984.

Department of the Army. "Ammunition and Explosives Safety Standards." AR 385-64, 22 May 1987.

Department of the Army. "Joint Munitions Effectiveness Manual of Fragmentation Data." FM 101-62-3.

Gilman, R. D. "Cased Granular Propellant Ignition." Date and source unknown.

Liddiard, T. P., and L. A. Roslund. "Projectile/Fragment Impact Sensitivity of Explosives." NSWC TR 89-184, Naval Surface Warfare Center, Silver Spring, MD, June 1993.

McClesky, F. "Quantity-Distance Fragment Hazard Computer Program (FRAGHAZ)." NSWC TR 87-59, Naval Surface Warfare Center, Silver Spring, MD, February 1988.

Victor, A. C. "Prediction/Analysis of Munition Reactions for Insensitive Munitions Threat Hazard Assessment." Proceedings of the Insensitive Munition Technology Symposium, Williamsburg, VA, June 1994.

INTENTIONALLY LEFT BLANK.

APPENDIX A:

PROPAGATION ACCIDENTS IN AMMUNITION HOLDING AREAS

INTENTIONALLY LEFT BLANK.

We considered several sources of information which are pertinent to propagation accidents in ammunition holding areas. These include reports of actual events found in the files of the Department of Defense Explosive Safety Board as well as results of hazard classification, packaging and MILVAN tests.

At least 57 accidents involving open storage of ammunition have been identified. We have reviewed reports of many of these. There are a number of difficulties associated with the data given in these reports. Much of it is old, dating to the World War II era. The observations were made at different times by different people with different ideas about what is important. The amount of information reported varies widely, and details of storage configurations were not usually included.

We drew a number of conclusions from our review. En masse detonation of large quantities of ammunition is rare, usually involving bombs or large projectiles. More commonly, propagation accidents follow a sequence of events involving fire in an ammunition stack and cookoff of munitions in that stack. This leads to propagation of fire to neighboring stacks via hot fragments and burning debris. The process is then repeated in events which may take days to unfold. Propulsive reactions in which items such as rockets and mortar rounds are launched toward neighboring or distant stacks often contribute to propagation. White phosphorus rounds are also major contributors to this type of propagation. Three illustrative accident reports are summarized in Table A-1.

The most relevant hazard classification test is the bonfire test. While this provides the maximum fragment radius, it doesn't always identify the number of fragments which travel shorter distances. Distances over which firebrands are spread are usually not reported. Another relevant series of tests were conducted in MILVAN containers.[1] Although the quantities of explosive in each MILVAN were modest, items were thrown large distances.

The importance of packaging was demonstrated by Teitell and Reeves[2] who conducted tests to determine the vulnerability of several munitions in wood packaging to several threats. The general results

---

[1] Lawrence, W. "Fragment Hazards From Munitions in Containers." BRL-TR-3203, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1991.

[2] Teitell, L., and H. J. Reeves. "Fire Retardant Packaging for Artillery Ammuniton." BRL-MR-2490, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1975.

Table A-1. Summary of Representative Propagation Accident Reports

DDESB FILE	466	480	1271
LOCATION	PUEBLO, CO	IE SHIMA, JAPAN	DA NANG, VIETNAM
AMMUNITION	3 "PILES" OF TNT-LOADED 3" M42 AMMUNITION SEPARATED BY 120' AND 285'	2162 TONS OF BOMBS, ROCKETS, SMALL ARMS, 20mm IN OPEN STORAGE WITH ABOUT 200' BETWEEN STACKS	LARGE MARINE CORPS AMMUNITION SUPPLY POINT
CAUSE	LIGHTNING		FIRE, POSSIBLY CAUSED BY ENEMY ACTION
DURATION	HOURS	2 DAYS	> 24 HOURS
RESULT	• MULTIPLE EVENTS • FIRE SPREAD BY HOT FRAGMENTS AND BURNING DEBRIS • VIRTUALY ALL AMMO LOST • FRAGS TO 3700' • THROWN ROUND DETONATED ON IMPACT AT 200' • BURNING WOOD AND FIBERBOARD TRAVELED 400'.	ESSENTIALLY EVERYTHING LOST	• MULTIPLE EXPLOSIONS • ROCKETS COMMUNICATED REACTION TO AN AIR FORCE ASP • LARGE LOSS OF AMMO • "NO CELL APPEARED TO COMMUNICATE DIRECTLY TO ITS NEIGHBOR."

of these tests were that the threats did not cause detonations but started fires in propellants and wood packaging. Propellant fires rapidly cooked off warheads and wood fires cooked off warheads causing both detonations and less violent explosions. Detonations sometimes scattered stacks and stopped reaction. Fire-resistant packaging prevented the spread of reaction in some cases.

We conclude that the dominant mode of propagation in ammunition accidents is fire leading to cookoff leading to more fire and more cookoff. Combustible packaging and propulsive reactions are major contributors to propagation. Insensitive munition technology will reduce the probability of propagation, especially where stacks can be separated by more that 50 ft. However, quantitative analysis of propagation probability for this mechanism is probably not possible at this time.

APPENDIX B:

FRAGPROP LISTING

INTENTIONALLY LEFT BLANK.

```
c*********************************************************************** F000010
c                                                                       F000020
      program frgpr                                                     F000030
c                                                                       F000040
c                              F R A G P R O P                          F000050
c                                                                       F000060
c      common blocks                                                    F000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   F000080
      common/repls/ nreps,fnreps                                        F000090
      common/frags/ nfrags,tfrags                                       F000100
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           F000110
      common/rkvar/ rkv(7)                                              F000120
c                                                                       F000130
c      type declarations                                                F000140
      character*8 uniela,unirem                                         F000150
      double precision rndinp,rndnxt,rndsav                             F000160
c                                                                       F000170
c      report the program title                                         F000180
      write(*,'(/,a)') ' FRAGPROP'                                      F000190
c                                                                       F000200
c      set up the run                                                   F000210
      call setup(rndinp,rndsav,rndnxt, ihbgn,tsbgn)                     F000220
c                                                                       F000230
c      begin the replication loop                                       F000240
      do 300 irep=1,nreps                                               F000250
c                                                                       F000260
        ifrcm=(irep-1)*nfrags                                           F000270
c                                                                       F000280
c       initialize the replication parameters                           F000290
        call rpini(rndnxt, slcnst,asldeg,alttud,wspeed,wdrdeg,wdrrad,   F000300
     +             jxrng,jrkmax)                                        F000310
c                                                                       F000320
c       begin the fragment loop                                         F000330
        do 200 ifrag=1,nfrags                                           F000340
c                                                                       F000350
c         estimate the remaining time                                   F000360
          frcmp=max(float(ifrcm+ifrag-1)/tfrags,smlnum)                 F000370
          call ertim(ihbgn,tsbgn,frcmp,timela,uniela,timrem,unirem)     F000380
c                                                                       F000390
c         report the current replication and                            F000400
c         fragment and the remaining time                               F000410
          write(*,'(a,i3,a,i3,a,i3,a,i3,a,f5.2,a)')                     F000420
     +    '+Replication: ',irep,'/',nreps,'   Fragment: ',ifrag,'/',    F000430
     +    nfrags,'   Remaining Time: ',timrem,unirem                    F000440
c                                                                       F000450
c         initialize the fragment parameters                            F000460
          call frini(irep,ifrag,rndnxt, irico,lrange,htfrft,mfrht1,     F000470
     +               vfrag,seldeg,selrad,aeldeg,aelrad,cd1,cd2,cd3,     F000480
     +               deldst)                                            F000490
c                                                                       F000500
c         begin the trajectory computation loop                         F000510
          itraj=0                                                       F000520
  100     itraj=itraj+1                                                 F000530
c                                                                       F000540
c           determine the time step                                     F000550
            call tstep(deldst,irico,vfrag,aelrad, deltim)               F000560
c                                                                       F000570
c           increment the time                                          F000580
            rkv(1)=rkv(1)+deltim                                        F000590
c                                                                       F000600
c           perform the Runge-Kutta integration                         F000610
            call rkint(wspeed,wdrdeg,wdrrad,jxrng,jrkmax,cd1,cd2,cd3,   F000620
     +                 deltim,alttud,ifrag)                             F000630
c                                                                       F000640
c           determine the fragment variable values                      F000650
c           at the end of the integration step                          F000660
            call frvar(vfrag,selrad,seldeg,aelrad,aeldeg,range,irange)  F000670
c                                                                       F000680
```

```
c            perform hazard computations as required                        F000690
             call hazch(ifrag,irange,mfrht1,mfrht2,lrange,aelrad,vfrag)     F000700
c                                                                           F000710
c            check the range for trajectory termination                     F000720
             if(range.gt.rngmax) go to 200                                  F000730
c                                                                           F000740
c            check the fragment velocity for trajectory termination         F000750
             if(vfrag.lt.20.0) go to 200                                    F000760
c                                                                           F000770
c            check for ricochet                                             F000780
             if(rkv(5).le.0.0) then                                         F000790
c                                                                           F000800
c              check the ricochet count                                     F000810
               if(irico.gt.6) go to 200                                     F000820
c                                                                           F000830
c              check the elevation angle for trajectory termination         F000840
               if(aeldeg.ge.asldeg) go to 200                               F000850
c                                                                           F000860
c              determine the ricochet velocity and angle                    F000870
               call ricoc(slcnst,vfrag,aeldeg,selrad,vrico,ricrad)          F000880
c                                                                           F000890
c              check the ricochet velocity for trajectory termination       F000900
               if(vrico.lt.20.0) go to 200                                  F000910
c                                                                           F000920
c              reinitialize the velocity integration variables              F000930
               call reini(irico,vrico,ricrad,wspeed,wdrrad, vfrag,          F000940
     +                    selrad,seldeg,aelrad,aeldeg,deldst,mfrht1)        F000950
c                                                                           F000960
             endif                                                          F000970
c                                                                           F000980
c          end of the trajectory computation loop                           F000990
           go to 100                                                        F001000
c                                                                           F001010
c        end of the fragment loop                                           F001020
  200    continue                                                           F001030
c                                                                           F001040
c        accumulate hit probabilities                                       F001050
         call hpacc                                                         F001060
c                                                                           F001070
c      end of the replication loop                                          F001080
  300  continue                                                           F001090
c                                                                           F001100
c      determine the elapsed time                                           F001110
       call ertim(ihbgn,tsbgn,1.0,timela,uniela,timrem,unirem)              F001120
c                                                                           F001130
c      write the output files                                               F001140
       call outpt                                                           F001150
c                                                                           F001160
c      report completion                                                    F001170
       write(*,'(a,f5.2,a/)')                                               F001180
     + '+Computation Complete                    Total Elapsed Time: ',     F001190
     + timela,uniela                                                        F001200
c                                                                           F001210
       stop ' '                                                             F001220
c                                                                           F001230
       end                                                                  F001240
c                                                                           F001250
c*************************************************************************** F001260
```

```
c*********************************************************************** S000010
c                                                                       S000020
      subroutine setup(rndinp,rndsav,rndnxt, ihbgn,tsbgn)               S000030
c                                                                       S000040
c      sets up the run                                                  S000050
c                                                                       S000060
c      common blocks                                                    S000070
      common/error/ meror,irerr,iferr,merout                            S000080
      common/rangs/ nrsegs,nrsegp                                       S000090
      common/denst/ fd(97),fdd(97),fdb(97),fdm(97)                      S000100
      common/phitt/ ph(97),phd(97),phb(97),phm(97)                      S000110
c                                                                       S000120
c      type declarations                                                S000130
      character dfile*25,afile*25                                       S000140
      double precision rndinp,rndnxt,rndsav                             S000150
c                                                                       S000160
c      assign constants                                                 S000170
      call const                                                        S000180
c                                                                       S000190
c      obtain input                                                     S000200
      call input(rndinp,rndsav,rndnxt,dfile,minvrt,afile)               S000210
c                                                                       S000220
c      compute parameters requiring both donor and acceptor input       S000230
      call doacc                                                        S000240
c                                                                       S000250
c      output the run conditions                                        S000260
      call outin(rndsav,dfile,minvrt,afile)                             S000270
c                                                                       S000280
c      compute the maximum vulnerable areas of the acceptor             S000290
      call vulna                                                        S000300
c                                                                       S000310
c      initialize the hit probabilities                                 S000320
      call arini(nrsegs,ph,phd,phb,phm,0.0)                             S000330
c                                                                       S000340
c      initialize the fragment densities                                S000350
      call arini(nrsegs,fd,fdd,fdb,fdm,0.0)                             S000360
c                                                                       S000370
c      obtain the starting time                                         S000380
      call gettim(ihbgn,imbgn,isbgn,i1bgn)                              S000390
      tsbgn=3600.0*float(ihbgn)+60.0*float(imbgn)                       S000400
     +     +float(isbgn)+float(i1bgn)/100.0                             S000410
c                                                                       S000420
c      report return                                                    S000430
      if(meror.gt.0) write(2,'(1x,a)') 'returning from setup'           S000440
c                                                                       S000450
      return                                                            S000460
c                                                                       S000470
      end                                                               S000480
c                                                                       S000490
c*********************************************************************** S000500
```

```
c*********************************************************************** C000010
c                                                                       C000020
      subroutine const                                                  C000030
c                                                                       C000040
c       assigns constants                                               C000050
c                                                                       C000060
c      common blocks                                                    C000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   C000080
      common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2,  C000090
     +              hcam3                                               C000100
      common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmtft,fi2tc2      C000110
c                                                                       C000120
c      mathematical constants                                           C000130
      qpi=atan(1.0)                                                     C000140
      hpi=2.0*qpi                                                       C000150
      pi=4.0*qpi                                                        C000160
      tpi=2.0*pi                                                        C000170
      raddeg=360.0/tpi                                                  C000180
      degrad=tpi/360.0                                                  C000190
      smlrad=1.0e-3*degrad                                              C000200
      bignum=1.0e30                                                     C000210
      smlnum=1.0e-30                                                    C000220
c                                                                       C000230
c      physical constants                                               C000240
      gg=32.174                                                         C000250
      tglbgr=1.4e4*gg                                                   C000260
      hrair0=3.8239e-2                                                  C000270
      racnst=3.39e4                                                     C000280
      vsound=1.1164e3                                                   C000290
      hccnst=2.86e5                                                     C000300
      hcam1=0.75                                                        C000310
      hcam2=1.5                                                         C000320
      hcam3=2.5                                                         C000330
c                                                                       C000340
c      unit conversions                                                 C000350
      fgrtlb=1./7.0e3                                                   C000360
      fgmtlb=2.2046226e-3                                               C000370
      fgrtgm=fgrtlb/fgmtlb                                              C000380
      ffttm=0.3048                                                      C000390
      ffttmm=1.e-3*ffttm                                                C000400
      fmtft=1.0/ffttm                                                   C000410
      fi2tc2=6.4516                                                     C000420
c                                                                       C000430
      return                                                            C000440
c                                                                       C000450
      end                                                               C000460
c                                                                       C000470
c*********************************************************************** C000480

c*********************************************************************** I000010
c                                                                       I000020
      subroutine input(rndinp,rndsav,rndnxt,dfile,minvrt,afile)         I000030
c                                                                       I000040
c       obtains input                                                   I000050
c                                                                       I000060
c       type declarations                                               I000070
      character dfile*25,afile*25                                       I000080
      double precision rndinp,rndnxt,rndsav                             I000090
c                                                                       I000100
c       input randomization seed                                        I000110
      call rndin(rndinp,rndsav,rndnxt)                                  I000120
c                                                                       I000130
c       input parameters                                                I000140
      call parin                                                        I000150
c                                                                       I000160
c       input donor data                                                I000170
      call donin(dfile,minvrt)                                          I000180
c                                                                       I000190
c       input acceptor data                                             I000200
      call accin(afile)                                                 I000210
c                                                                       I000220
c       input range data                                                I000230
      call rngin                                                        I000240
c                                                                       I000250
      return                                                            I000260
c                                                                       I000270
      end                                                               I000280
c                                                                       I000290
c*********************************************************************** I000300
```

```
c*********************************************************************** R000010
c                                                                       R000020
      subroutine rndin(rndinp,rndsav,rndnxt)                            R000030
c                                                                       R000040
c      obtains randomization seed input                                 R000050
c                                                                       R000060
c      type declarations                                                R000070
      double precision rndinp,rndsav,rndnxt                             R000080
c                                                                       R000090
  100 write(*,'(a\)') ' Enter Monte Carlo Seed (1 to 2147483646): '     R000100
      read(*,*,err=100) rndinp                                          R000110
      if(rndinp.lt.1.0.or.rndinp.gt.2147483646.0) go to 100             R000120
c                                                                       R000130
      rndsav=rndinp                                                     R000140
      rnddum=rndom(rndinp)                                              R000150
      rndnxt=0.0d0                                                      R000160
c                                                                       R000170
      return                                                            R000180
c                                                                       R000190
      end                                                               R000200
c                                                                       R000210
c*********************************************************************** R000220

c*********************************************************************** P000010
c                                                                       P000020
      subroutine parin                                                  P000030
c                                                                       P000040
c      obtains parameter input                                          P000050
c                                                                       P000060
c      common blocks                                                    P000070
      common/error/ meror,irerr,iferr,merout                            P000080
      common/repls/ nreps,fnreps                                        P000090
      common/prmtr/ slcmin,slcmax,altmin,altmax,                        P000100
     +              wspmin,wspmax,wdrmin,wdrmax                         P000110
c                                                                       P000120
      open(4,file='fp.prm',status='old')                                P000130
      read(4,*) meror,irerr,iferr                                       P000140
      if(meror.gt.0) open(2,file='fp.err',status='unknown')             P000150
      read(4,*) nreps                                                   P000160
      nreps=min(nreps,999)                                              P000170
      fnreps=float(nreps)                                               P000180
      read(4,*) slcmin,slcmax                                           P000190
      read(4,*) altmin,altmax                                           P000200
      read(4,*) wspmin,wspmax                                           P000210
      read(4,*) wdrmin,wdrmax                                           P000220
      close(4)                                                          P000230
c                                                                       P000240
      return                                                            P000250
c                                                                       P000260
      end                                                               P000270
c                                                                       P000280
c*********************************************************************** P000290
```

```
c*********************************************************************** D000010
c                                                                       D000020
      subroutine donin(dfile,minvrt)                                    D000030
c                                                                       D000040
c      obtains donor input                                              D000050
c                                                                       D000060
c      common blocks                                                    D000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   D000080
      common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2,  D000090
     +              hcam3                                               D000100
      common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmtft,fi2tc2      D000110
      common/error/ meror,irerr,iferr,merout                            D000120
      common/frmin/ fmgrmn,fmgmmn,ezndeg                                D000130
      common/zones/ elzdeg,azsdeg,azsrad                                D000140
      common/donor/ nunits,htstft,htbaft                                D000150
      common/frags/ nfrags,tfrags                                       D000160
      common/fragd/ famil(900),vfifs(900),farma(900),                   D000170
     +              fatn(900),ezldeg(900)                               D000180
      common/fragc/ facm2(900),fdmm(900),fmgm(900)                      D000190
      common/fragx/ vcrdx(900),vcrbx(900)                               D000200
      common/fragk/ frkec(900)                                          D000210
      common/names/ dname,aname                                         D000220
c                                                                       D000230
c      local array                                                      D000240
      dimension fmult(36)                                               D000250
c                                                                       D000260
c      type declarations                                                D000270
      character dname*40,aname*40                                       D000280
      character dfile*25,answr*4                                        D000290
c                                                                       D000300
c      report call                                                      D000310
      if(meror.gt.0) write(2,'(4x,a)') 'donin called'                   D000320
c                                                                       D000330
c      user specification input                                         D000340
      open(4,file='fp.dnr',status='old')                                D000350
      read(4,'(40x,a)') dname                                           D000360
      read(4,'(40x,a)') dfile                                           D000370
      read(4,'(40x,a)') answr                                           D000380
      minvrt=0                                                          D000390
      if(answr.eq.'Y') minvrt=1                                         D000400
      if(answr.eq.'y') minvrt=1                                         D000410
      if(answr.eq.'YES') minvrt=1                                       D000420
      if(answr.eq.'Yes') minvrt=1                                       D000430
      if(answr.eq.'yes') minvrt=1                                       D000440
      read(4,'(bn,40x,i5)') nunits                                      D000450
      read(4,'(bn,40x,f10.2)') htstft                                   D000460
      read(4,'(bn,40x,f10.2)') htbaft                                   D000470
      read(4,'(bn,40x,f10.2)') fmgrmn                                   D000480
      fmgmmn=fgrtgm*fmgrmn                                              D000490
      read(4,'(bn,40x,f10.2)') ezndeg                                   D000500
      close(4)                                                          D000510
c                                                                       D000520
c      donor lethality input                                            D000530
      open(4,file=dfile,status='old')                                   D000540
      read(4,*) elzdeg,azsdeg                                           D000550
      azsrad=degrad*azsdeg                                              D000560
      read(4,*) nfmlts,mdosh                                            D000570
```

```
      if(mdosh.gt.0) then                                                D000580
        read(4,*) dshin                                                  D000590
        dshcm=2.54*dshin                                                 D000600
        read(4,*) ccdsv,athdsv,bthdsv,cthdsv                             D000610
        tcdsv=10.0**ccdsv                                                D000620
        read(4,*) ccdsm,athdsm,bthdsm,cthdsm                             D000630
        tcdsm=10.0**ccdsm                                                D000640
      endif                                                              D000650
      read(4,*) (fmult(ifmlt),ifmlt=1,nfmlts)                            D000660
      fnuaz=float(nunits)*azsdeg                                         D000670
      ifrag=1                                                            D000680
  100 read(4,*,end=200) poldeg,fmgr,tfamil,tvfifs,tfarma                 D000690
      if(minvrt.eq.0) then                                               D000700
        tezdeg=90.0-poldeg                                               D000710
      else                                                               D000720
        tezdeg=poldeg-90.0                                               D000730
      endif                                                              D000740
      if(tezdeg.lt.ezndeg) go to 100                                     D000750
      tfmgm=0.06479891*fmgr                                              D000760
      tfacm2=fi2tc2*tfamil*fgrtlb*fmgr                                   D000770
      if(mdosh.gt.0) then                                                D000780
        call vmthr(dshcm,tfacm2,tvfifs,tfmgm,tcdsv,athdsv,bthdsv,cthdsv, D000790
     +             tcdsm,athdsm,bthdsm,cthdsm,tvfifs,tfmgm)              D000800
        if(tvfifs.le.0.0) go to 100                                      D000810
      endif                                                              D000820
      if(tfmgm.lt.fmgmmn) go to 100                                      D000830
      if(ifrag.gt.900) then                                              D000840
        write(*,'(a,i3,a)')                                              D000850
     +   ' Data file contains more than ',900,' fragments!'              D000860
        stop 'Computation Terminated'                                    D000870
      endif                                                              D000880
      ezldeg(ifrag)=tezdeg                                               D000890
      fmgm(ifrag)=tfmgm                                                  D000900
      famil(ifrag)=tfamil                                                D000910
      vfifs(ifrag)=tvfifs                                                D000920
      farma(ifrag)=tfarma                                                D000930
      fatn(ifrag)=fnuaz*fmult(int(poldeg/elzdeg))                        D000940
      facm2(ifrag)=tfacm2                                                D000950
      fdmm(ifrag)=2.0*sqrt(100.0*tfacm2/pi)                              D000960
      frkec(ifrag)=fmgr/tglbgr                                           D000970
      ifrag=ifrag+1                                                      D000980
      go to 100                                                          D000990
  200 nfrags=ifrag-1                                                     D001000
      close(4)                                                           D001010
c                                                                        D001020
c      report return                                                     D001030
      if(meror.gt.0) write(2,'(4x,a)') 'returning from donin'            D001040
c                                                                        D001050
      return                                                             D001060
c                                                                        D001070
      end                                                                D001080
c                                                                        D001090
c*************************************************************************D001100
```

```
c*********************************************************************** A000010
c                                                                       A000020
      subroutine accin(afile)                                           A000030
c                                                                       A000040
c      obtains acceptor input                                           A000050
c                                                                       A000060
c      common blocks                                                    A000070
      common/error/ meror,irerr,iferr,merout                            A000080
      common/names/ dname,aname                                         A000090
      common/acptr/ htacft,wdacft,dpacft,htacfm                         A000100
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       A000110
      common/geowp/ wplin,wpdin                                         A000120
      common/geopa/ pasin,pahin,pasft                                   A000130
      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin                 A000140
      common/jrdtx/ abjrx,cjrx                                          A000150
      common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw                       A000160
      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin                 A000170
      common/jrdtp/ abjrp,cjrp                                          A000180
      common/thorr/ athr,bthr,cthr,dthr,vrrms,tcr                       A000190
      common/geoas/ ashcm                                               A000200
      common/thras/ athasv,bthasv,cthasv,tcasv,                         A000210
     +              athasm,bthasm,cthasm,tcasm                          A000220
      common/mechd/ wpke                                                A000230
c                                                                       A000240
c      type declarations                                                A000250
      character dname*40,aname*40                                       A000260
      character afile*25                                                A000270
c                                                                       A000280
c      report call                                                      A000290
      if(meror.gt.0) write(2,'(4x,a)') 'accin called'                   A000300
c                                                                       A000310
c      user specification input                                         A000320
      open(4,file='fp.acc',status='old')                                A000330
      read(4,'(40x,a)') aname                                           A000340
      read(4,'(40x,a)') afile                                           A000350
      read(4,'(bn,40x,f10.2)') htacft                                   A000360
      htacfm=htacft-0.2                                                 A000370
      read(4,'(bn,40x,f10.2)') wdacft                                   A000380
      read(4,'(bn,40x,f10.2)') dpacft                                   A000390
      close(4)                                                          A000400
c                                                                       A000410
c      acceptor vulnerability input                                     A000420
      open(4,file=afile,status='old')                                   A000430
      read(4,*) nweps,macwh,macrm,macsh,mstor                           A000440
      read(4,*) wplin,wpdin                                             A000450
      read(4,*) pasin,pahin                                             A000460
      pasft=pasin/12.                                                   A000470
      if(macwh.ne.0) then                                               A000480
        read(4,*) whlin,whdin,whhin,exlin                               A000490
        whhcm=2.54*whhin                                                A000500
        whhmm=10.0*whhcm                                                A000510
        read(4,*) ajrx,bjrxp1,cjrx                                      A000520
        abjrx=ajrx*bjrxp1                                               A000530
        read(4,*) ccw,athw,bthw,cthw,dthw,vrwms                         A000540
        tcw=10.0**ccw                                                   A000550
      endif                                                             A000560
      if(macrm.ne.0) then                                               A000570
        read(4,*) rmlin,rmdin,rmhin,prlin                               A000580
        rmhcm=2.54*rmhin                                                A000590
        rmhmm=10.0*rmhcm                                                A000600
        read(4,*) ajrp,bjrpp1,cjrp                                      A000610
        abjrp=ajrp*bjrpp1                                               A000620
```

```
        read(4,*) ccr,athr,bthr,cthr,dthr,vrrms                          A000630
        tcr=10.0**ccr                                                    A000640
      endif                                                              A000650
      if(macsh.ne.0) then                                                A000660
        read(4,*) ashin                                                  A000670
        ashcm=2.54*ashin                                                 A000680
        read(4,*) ccasv,athasv,bthasv,cthasv                             A000690
        tcasv=10.0**ccasv                                                A000700
        read(4,*) ccasm,athasm,bthasm,cthasm                             A000710
        tcasm=10.0**ccasm                                                A000720
      endif                                                              A000730
      read(4,*) wpke                                                     A000740
      close(4)                                                           A000750
c                                                                        A000760
c      report return                                                     A000770
      if(meror.gt.0) write(2,'(4x,a)') 'returning from accin'            A000780
c                                                                        A000790
      return                                                             A000800
c                                                                        A000810
      end                                                                A000820
c                                                                        A000830
c*********************************************************************** A000840

c*********************************************************************** R000010
c                                                                        R000020
      subroutine rngin                                                   R000030
c                                                                        R000040
c      obtains range input                                               R000050
c                                                                        R000060
c      common blocks                                                     R000070
      common/error/ meror,irerr,iferr,merout                             R000080
      common/rangs/ nrsegs,nrsegp                                        R000090
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin            R000100
      common/arang/ rangem(97),ahfrn(97),ahtop(97)                       R000110
c                                                                        R000120
c      report call                                                       R000130
      if(meror.gt.0) write(2,'(2x,a)') 'rngin called'                    R000140
c                                                                        R000150
c      user specification input                                          R000160
      open(4,file='fp.rng',status='old')                                 R000170
      read(4,'(bn,40x,f10.2)') rngmax                                    R000180
      read(4,'(bn,40x,i5)') nrsegs                                       R000190
      nrsegs=min(nrsegs,96)                                              R000200
      nrsegp=nrsegs+1                                                    R000210
      rngseg=rngmax/float(nrsegs)                                        R000220
      rngsgh=0.5*rngseg                                                  R000230
      rngscl=1.0e-2*rngseg                                               R000240
      rngsix=0.6*rngseg                                                  R000250
      do 100 irange=1,nrsegp                                             R000260
        rangem(irange)=rngseg*(float(irange)-0.5)                        R000270
  100 continue                                                           R000280
      close(4)                                                           R000290
c                                                                        R000300
c      report return                                                     R000310
      if(meror.gt.0) write(2,'(4x,a)') 'returning from rngin'            R000320
c                                                                        R000330
      return                                                             R000340
c                                                                        R000350
      end                                                                R000360
c                                                                        R000370
c*********************************************************************** R000380
```

```
c*********************************************************************** D000010
c                                                                       D000020
      subroutine doacc                                                  D000030
c                                                                       D000040
c      computes parameters requiring both donor and acceptor input      D000050
c                                                                       D000060
c      common blocks                                                    D000070
      common/error/ meror,irerr,iferr,merout                            D000080
      common/repls/ nreps,fnreps                                        D000090
      common/zones/ elzdeg,azsdeg,azsrad                                D000100
      common/frags/ nfrags,tfrags                                       D000110
      common/fragd/ famil(900),vfifs(900),farma(900),                   D000120
     +              fatn(900),ezldeg(900)                               D000130
      common/fragc/ facm2(900),fdmm(900),fmgm(900)                      D000140
      common/fragx/ vcrdx(900),vcrbx(900)                               D000150
      common/fragp/ vcrdp(900),vcrbp(900)                               D000160
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       D000170
      common/acptr/ htacft,wdacft,dpacft,htacfm                         D000180
      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin                 D000190
      common/jrdtx/ abjrx,cjrx                                          D000200
      common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw                       D000210
      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin                 D000220
      common/thorr/ athr,bthr,cthr,dthr,vrrms,tcr                       D000230
      common/jrdtp/ abjrp,cjrp                                          D000240
      common/rangs/ nrsegs,nrsegp                                       D000250
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           D000260
      common/arang/ rangem(97),anfrn(97),ahtop(97)                      D000270
c                                                                       D000280
c      report call                                                      D000290
      if(meror.gt.0) write(2,'(2x,a)') 'doacc called'                   D000300
c                                                                       D000310
c      compute the total number of fragment trajectories                D000320
      tfrags=float(nreps*nfrags)                                        D000330
c                                                                       D000340
c      compute the minimum range and range area arrays                  D000350
      tahazs=tan(0.5*azsrad)                                            D000360
      if(tahazs.gt.0.0) then                                            D000370
        rngmin=wdacft/(2.0*tahazs)                                      D000380
      else                                                              D000390
        rngmin=0.0                                                      D000400
      endif                                                             D000410
      do 100 irange=1,nrsegp                                            D000420
        rangeo=rangem(irange)+rngsgh                                    D000430
        rangei=rangem(irange)-rngsgh                                    D000440
        ahfrn(irange)=azsrad*htacft*rangem(irange)                      D000450
        ahtop(irange)=0.5*azsrad*(rangeo*rangeo-rangei*rangei)          D000460
  100 continue                                                          D000470
c                                                                       D000480
c      predetermine parameters characterizing                           D000490
c      fragment lethality against unshrouded acceptor                   D000500
      if(macsh.eq.0) then                                               D000510
c                                                                       D000520
        if(macwh.ne.0) then                                             D000530
c                                                                       D000540
c          a warhead component is present                               D000550
          do 200 ifrag=1,nfrags                                         D000560
            vcrdx(ifrag)=vcrjr(abjrx,cjrx,whhmm,fdmm(ifrag))            D000570
            vcrbx(ifrag)=                                               D000580
     +        vcrth(tcw,athw,bthw,whhcm,facm2(ifrag),fmgm(ifrag))       D000590
  200     continue                                                      D000600
c                                                                       D000610
        endif                                                           D000620
c                                                                       D000630
        if(macrm.ne.0) then                                             D000640
c                                                                       D000650
c          a rocket motor component is present                          D000660
          do 300 ifrag=1,nfrags                                         D000670
            vcrdp(ifrag)=vcrjr(abjrp,cjrp,whhmm,fdmm(ifrag))            D000680
            vcrbp(ifrag)=                                               D000690
     +        vcrth(tcr,athr,bthr,rmhcm,facm2(ifrag),fmgm(ifrag))       D000700
  300     continue                                                      D000710
c                                                                       D000720
        endif                                                           D000730
c                                                                       D000740
      endif                                                             D000750
c                                                                       D000760
c      report return                                                    D000770
      if(meror.gt.0) write(2,'(2x,a)') 'returning from doacc'           D000780
c                                                                       D000790
      return                                                            D000800
c                                                                       D000810
      end                                                               D000820
c                                                                       D000830
c*********************************************************************** D000840
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulna                                                  V000030
c                                                                       V000040
c      computes the maximum vulnerable areas of the acceptor            V000050
c                                                                       V000060
c      common blocks                                                    V000070
      common/error/ meror,irerr,iferr,merout                            V000080
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       V000090
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 V000100
c                                                                       V000110
c      report call                                                      V000120
      if(meror.gt.0) write(2,'(2x,a)') 'vulna called'                   V000130
c                                                                       V000140
c      initialize the vulnerable areas                                  V000150
      amfrn=0.0                                                         V000160
      amtop=0.0                                                         V000170
      axfrn=0.0                                                         V000180
      axtop=0.0                                                         V000190
      apfrn=0.0                                                         V000200
      aptop=0.0                                                         V000210
c                                                                       V000220
      if(mstor.eq.1) then                                               V000230
c                                                                       V000240
c        the storage arrangement is vertical                            V000250
        call vulnv(ucfrn,uctop)                                         V000260
c                                                                       V000270
c        if a warhead component is present                              V000280
        if(macwh.ne.0) call vulvw(ucfrn,uctop)                          V000290
c                                                                       V000300
c        if a rocket motor component is present                         V000310
        if(macrm.ne.0) call vulvm(ucfrn,uctop)                          V000320
c                                                                       V000330
      else                                                              V000340
c                                                                       V000350
c        the storage arrangement is horizontal                          V000360
        call vulnh(ucfrn,uctop)                                         V000370
c                                                                       V000380
        amin2=0.0                                                       V000390
c                                                                       V000400
c        if a warhead component is present                              V000410
        if(macwh.ne.0) call vulhw(amin2,ucfrn,uctop)                    V000420
c                                                                       V000430
c        if a rocket motor component is present                         V000440
        if(macrm.ne.0) call vulhm(amin2,ucfrn,uctop)                    V000450
c                                                                       V000460
        amfrn=ucfrn*amin2                                               V000470
        amtop=uctop*amin2                                               V000480
c                                                                       V000490
      endif                                                             V000500
c                                                                       V000510
c      report return                                                    V000520
      if(meror.gt.0) write(2,'(2x,a)') 'returning from vulna'           V000530
c                                                                       V000540
      return                                                            V000550
c                                                                       V000560
      end                                                               V000570
c                                                                       V000580
c*********************************************************************** V000590
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulnv(ucfrn,uctop)                                     V000030
c                                                                       V000040
c      determines the number of front and top                           V000050
c      units for a vertical storage arrangement                         V000060
c                                                                       V000070
c      common blocks                                                    V000080
      common/error/ meror,irerr,iferr,merout                            V000090
      common/acptr/ htacft,wdacft,dpacft,htacfm                         V000100
      common/geowp/ wplin,wpdin                                         V000110
      common/geopa/ pasin,pahin,pasft                                   V000120
c                                                                       V000130
c      report call                                                      V000140
      if(meror.gt.0) write(2,'(4x,a)') 'vulnv called'                   V000150
c                                                                       V000160
c      compute the front and top unit areas in sq in                    V000170
      wdpps=wpdin+pasin                                                 V000180
      aufrn=(wplin+pahin)*wdpps                                         V000190
      autop=wdpps*wdpps                                                 V000200
c                                                                       V000210
c      compute the numbers of front and                                 V000220
c      top units (with conversion to sq ft)                             V000230
      wdacp=wdacft+pasft                                                V000240
      ucfrn=wdacp*htacft/aufrn                                          V000250
      uctop=wdacp*(dpacft+pasft)/autop                                  V000260
c                                                                       V000270
c      report return                                                    V000280
      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulnv'           V000290
c                                                                       V000300
      return                                                            V000310
c                                                                       V000320
      end                                                               V000330
c                                                                       V000340
c*********************************************************************** V000350

c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulvw(ucfrn,uctop)                                     V000030
c                                                                       V000040
c      determines the front and top vulnerable                          V000050
c      warhead areas for vertical storage                               V000060
c                                                                       V000070
c      common blocks                                                    V000080
      common/error/ meror,irerr,iferr,merout                            V000090
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   V000100
      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin                 V000110
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 V000120
c                                                                       V000130
c      report call                                                      V000140
      if(meror.gt.0) write(2,'(4x,a)') 'vulvw called'                   V000150
c                                                                       V000160
c      compute the front and top maximum                                V000170
c      vulnerable warhead areas in sq ft                                V000180
      amfrn=ucfrn*whlin*whdin                                           V000190
      amtop=uctop*qpi*whdin*whdin                                       V000200
c                                                                       V000210
c      compute the front and top maximum                                V000220
c      vulnerable explosive areas in sq ft                              V000230
      exdin=whdin-whhin                                                 V000240
      axfrn=ucfrn*exlin*exdin                                           V000250
      axtop=uctop*qpi*exdin*exdin                                       V000260
c                                                                       V000270
c      report return                                                    V000280
      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulvw'           V000290
c                                                                       V000300
      return                                                            V000310
c                                                                       V000320
      end                                                               V000330
c                                                                       V000340
c*********************************************************************** V000350
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulvm(ucfrn,uctop)                                     V000030
c                                                                       V000040
c      determines the front and top vulnerable                          V000050
c      rocket motor areas for vertical storage                          V000060
c      (there are no top vulnerable areas if a warhead is also present) V000070
c                                                                       V000080
c      common blocks                                                    V000090
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   V000100
      common/error/ meror,irerr,iferr,merout                            V000110
      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin                 V000120
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 V000130
c                                                                       V000140
c      report call                                                      V000150
      if(meror.gt.0) write(2,'(4x,a)') 'vulvm called'                   V000160
c                                                                       V000170
c      compute the front and top maximum                                V000180
c      vulnerable motor areas in sq ft                                  V000190
      amfrn=amfrn+ucfrn*rmlin*rmdin                                     V000200
      if(amtop.eq.0.0) amtop=uctop*qpi*rmdin*rmdin                      V000210
c                                                                       V000220
c      compute the front and top maximum                                V000230
c      vulnerable propellant areas in sq ft                             V000240
      prdin=rmdin-rmhin                                                 V000250
      apfrn=ucfrn*prlin*prdin                                           V000260
      if(axtop.eq.0.0) aptop=uctop*qpi*prdin*prdin                      V000270
c                                                                       V000280
c      report return                                                    V000290
      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulvm'           V000300
c                                                                       V000310
      return                                                            V000320
c                                                                       V000330
      end                                                               V000340
c                                                                       V000350
c*********************************************************************** V000360

c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulnh(ucfrn,uctop)                                     V000030
c                                                                       V000040
c      determines the number of front and top units for a               V000050
c      horizontal storage arrangement (nweps weapons high)              V000060
c                                                                       V000070
c      common blocks                                                    V000080
      common/error/ meror,irerr,iferr,merout                            V000090
      common/acptr/ htacft,wdacft,dpacft,htacfm                         V000100
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       V000110
      common/geowp/ wplin,wpdin                                         V000120
      common/geopa/ pasin,pahin,pasft                                   V000130
c                                                                       V000140
c      report call                                                      V000150
      if(meror.gt.0) write(2,'(4x,a)') 'vulnh called'                   V000160
c                                                                       V000170
c      compute the top and front unit areas in sq in                    V000180
      autop=wplin*(wpdin+pasin)                                         V000190
      aufrn=autop+wplin*pahin/float(nweps)                              V000200
c                                                                       V000210
c      compute the numbers of front and                                 V000220
c      top units (with conversion to sq ft)                             V000230
      ucfrn=wdacft*(htacft+pasft)/aufrn                                 V000240
      uctop=wdacft*(dpacft+pasft)/autop                                 V000250
c                                                                       V000260
c      report return                                                    V000270
      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulnh'           V000280
c                                                                       V000290
      return                                                            V000300
c                                                                       V000310
      end                                                               V000320
c                                                                       V000330
c*********************************************************************** V000340
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulhw(amin2,ucfrn,uctop)                               V000030
c                                                                       V000040
c      determines the front and top vulnerable                          V000050
c      warhead areas for horizontal storage                             V000060
c                                                                       V000070
c      common blocks                                                    V000080
      common/error/ meror,irerr,iferr,merout                            V000090
      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin                 V000100
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 V000110
c                                                                       V000120
c      report call                                                      V000130
      if(meror.gt.0) write(2,'(4x,a)') 'vulhw called'                   V000140
c                                                                       V000150
c      compute the front and top maximum                                V000160
c      vulnerable warhead areas in sq ft                                V000170
      amin2=whlin*whdin                                                 V000180
c                                                                       V000190
c      compute the front and top maximum                                V000200
c      vulnerable explosive areas in sq ft                              V000210
      axin2=(whdin-whhin)*exlin                                         V000220
      axfrn=ucfrn*axin2                                                 V000230
      axtop=uctop*axin2                                                 V000240
c                                                                       V000250
c      report return                                                    V000260
      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulhw'           V000270
c                                                                       V000280
      return                                                            V000290
c                                                                       V000300
      end                                                               V000310
c                                                                       V000320
c*********************************************************************** V000330

c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulhm(amin2,ucfrn,uctop)                               V000030
c                                                                       V000040
c      determines the front and top vulnerable                          V000050
c      rocket motor areas for horizontal storage                        V000060
c                                                                       V000070
c      common blocks                                                    V000080
      common/error/ meror,irerr,iferr,merout                            V000090
      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin                 V000100
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 V000110
c                                                                       V000120
c      report call                                                      V000130
      if(meror.gt.0) write(2,'(4x,a)') 'vulhm called'                   V000140
c                                                                       V000150
c      compute the front and top maximum                                V000160
c      vulnerable motor areas in sq ft                                  V000170
      amin2=amin2+rmlin*rmdin                                           V000180
c                                                                       V000190
c      compute the front and top maximum                                V000200
c      vulnerable propellant areas in sq ft                             V000210
      apin2=(rmdin-rmhin)*prlin                                         V000220
      apfrn=apfrn+ucfrn*apin2                                           V000230
      aptop=aptop+uctop*apin2                                           V000240
c                                                                       V000250
c      report return                                                    V000260
      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulhm'           V000270
c                                                                       V000280
      return                                                            V000290
c                                                                       V000300
      end                                                               V000310
c                                                                       V000320
c*********************************************************************** V000330
```

```
c*********************************************************************** O0000010
c                                                                       O0000020
      subroutine outin(rndsav,dfile,minvrt,afile)                       O0000030
c                                                                       O0000040
c      outputs problem input                                            O0000050
c                                                                       O0000060
c      common blocks                                                    O0000070
      common/repls/ nreps,fnreps                                        O0000080
      common/frmin/ fmgrmn,fmgmmn,ezndeg                                O0000090
      common/zones/ elzdeg,azsdeg,azsrad                                O0000100
      common/prmtr/ slcmin,slcmax,altmin,altmax,                        O0000110
     +              wspmin,wspmax,wdrmin,wdrmax                         O0000120
      common/donor/ nunits,htstft,htbaft                                O0000130
      common/frags/ nfrags,tfrags                                       O0000140
      common/rangs/ nrsegs,nrsegp                                       O0000150
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           O0000160
      common/names/ dname,aname                                         O0000170
      common/acptr/ htacft,wdacft,dpacft,htacfm                         O0000180
c                                                                       O0000190
c      type declarations                                                O0000200
      character dname*40,aname*40                                       O0000210
      character dfile*25,afile*25                                       O0000220
      double precision rndsav                                           O0000230
c                                                                       O0000240
      open(3,file='fp.out',status='unknown')                            O0000250
c                                                                       O0000260
      write(3,'(//33x,a)') '***FRAGPROP***'                             O0000270
      write(3,'( /14x,a)')                                              O0000280
     + 'REACTION PROPAGATION PROGRAM FOR FRAGMENTING MUNITIONS'         O0000290
c                                                                       O0000300
      write(3,'(//33x,a/)')     '**PARAMETERS**'                        O0000310
      call inout(' NUMBER OF REPLICATIONS: ',nreps)                     O0000320
      call flout('  MINIMUM SOIL CONSTANT: ',slcmin,'       ')          O0000330
      call flout('  MAXIMUM SOIL CONSTANT: ',slcmax,'       ')          O0000340
      call flout('       MINIMUM ALTITUDE: ',altmin,' ft    ')          O0000350
      call flout('       MAXIMUM ALTITUDE: ',altmax,' ft    ')          O0000360
      call flout('     MINIMUM WIND SPEED: ',wspmin,' mph   ')          O0000370
      call flout('     MAXIMUM WIND SPEED: ',wspmax,' mph   ')          O0000380
      call flout(' MINIMUM WIND DIRECTION: ',wdrmin,' degrees')         O0000390
      call flout(' MAXIMUM WIND DIRECTION: ',wdrmax,' degrees')         O0000400
c                                                                       O0000410
      write(3,'(//32x,a/)')     '**RANDOMIZATION**'                     O0000420
      write(3,'(24x,a,i10)')      '   MONTE CARLO SEED: ',idint(rndsav) O0000430
c                                                                       O0000440
      write(3,'(//36x,a/)')     '**DONOR**'                             O0000450
      write(3,'(24x,a,a)')        '        DESCRIPTION: ',dname       O0000460
      write(3,'(24x,a,a)')        'LETHALITY DATA FILE: ',dfile       O0000470
      if(minvrt.eq.0) then                                              O0000480
        write(3,'(24x,a)')        '         ORIENTATION: Erect'       O0000490
      else                                                              O0000500
        write(3,'(24x,a)')        '         ORIENTATION: Inverted'    O0000510
      endif                                                             O0000520
      call inout('         NUMBER OF UNITS: ',nunits)                   O0000530
      call flout('         HEIGHT OF STACK: ',htstft,' ft     ')        O0000540
      call flout('          BASE ELEVATION: ',htbaft,' ft     ')        O0000550
      call inout('    NUMBER OF FRAGMENTS: ',nfrags)                    O0000560
      call flout('  MINIMUM FRAGMENT MASS: ',fmgrmn,' grains ')        O0000570
      call flout('MINIMUM ELEVATION ANGLE: ',ezndeg,' degrees')         O0000580
c                                                                       O0000590
      write(3,'(//36x,a/)')     '**RANGE**'                             O0000600
      call flout('          MINIMUM RANGE: ',rngmin,' ft     ')        O0000610
      call flout('          MAXIMUM RANGE: ',rngmax,' ft     ')        O0000620
      call inout('     NUMBER OF SEGMENTS: ',nrsegs)                    O0000630
c                                                                       O0000640
      write(3,'(//34x,a/)')   '**ACCEPTOR**'                            O0000650
      write(3,'(20x,a,a)')        '        DESCRIPTION: ',aname       O0000660
      write(3,'(20x,a,a)')      'VULNERABILITY DATA FILE: ',afile     O0000670
      call flout('         HEIGHT OF STACK: ',htacft,' ft    ')         O0000680
      call flout('          WIDTH OF STACK: ',wdacft,' ft    ')         O0000690
      call flout('          DEPTH OF STACK: ',dpacft,' ft    ')         O0000700
c                                                                       O0000710
      close(3)                                                          O0000720
c                                                                       O0000730
      return                                                            O0000740
c                                                                       O0000750
      end                                                               O0000760
c                                                                       O0000770
c*********************************************************************** O0000780
```

```
c*********************************************************************** F000010
c                                                                       F000020
      subroutine flout(alabel,flvar,aunit)                              F000030
c                                                                       F000040
c      outputs a floating point variable                                F000050
c      with an alphanumeric label and units                             F000060
c                                                                       F000070
c      type declarations                                                F000080
      character alabel*25,aunit*8                                       F000090
c                                                                       F000100
      aflvar=abs(flvar)                                                 F000110
      if(aflvar.lt.1.0) then                                            F000120
        if(flvar.lt.0.0) then                                           F000130
          write(3,'(20x,2a,f2.1,a)') alabel,'  -0',aflvar,aunit         F000140
        else                                                            F000150
          write(3,'(20x,2a,f2.1,a)') alabel,'   0',flvar,aunit          F000160
        endif                                                           F000170
      else                                                              F000180
        write(3,'(20x,a,f6.1,a)') alabel,flvar,aunit                    F000190
      endif                                                             F000200
c                                                                       F000210
      return                                                            F000220
c                                                                       F000230
      end                                                               F000240
c                                                                       F000250
c*********************************************************************** F000260

c*********************************************************************** I000010
c                                                                       I000020
      subroutine inout(alabel,invar)                                    I000030
c                                                                       I000040
c      outputs an integer variable with an alphanumeric label           I000050
c                                                                       I000060
c      type declaration                                                 I000070
      character alabel*25                                               I000080
c                                                                       I000090
      write(3,'(20x,a,i4)') alabel,invar                                I000100
c                                                                       I000110
      return                                                            I000120
c                                                                       I000130
      end                                                               I000140
c                                                                       I000150
c*********************************************************************** I000160

c*********************************************************************** A000010
c                                                                       A000020
      subroutine arini(nrange,v,vd,vb,vm,value)                         A000030
c                                                                       A000040
c      initializes variable arrays                                      A000050
c                                                                       A000060
c      local arrays                                                     A000070
      dimension v(97),vd(97),vb(97),vm(97)                              A000080
c                                                                       A000090
      do 100 irange=1,nrange                                            A000100
        v (irange)=value                                                A000110
        vd(irange)=value                                                A000120
        vb(irange)=value                                                A000130
        vm(irange)=value                                                A000140
  100 continue                                                          A000150
c                                                                       A000160
      return                                                            A000170
c                                                                       A000180
      end                                                               A000190
c                                                                       A000200
c*********************************************************************** A000210
```

```
c*********************************************************************** E000010
c                                                                       E000020
      subroutine ertim(ihbgn,tsbgn,frcmp,timela,uniela,timrem,unirem)   E000030
c                                                                       E000040
c     determines the elapsed time and estimates the remaining time      E000050
c                                                                       E000060
c     type declaration                                                  E000070
      character*8 uniela,unirem                                         E000080
c                                                                       E000090
c     obtain the current time                                           E000100
      call gettim(ihnow,imnow,isnow,ilnow)                              E000110
c                                                                       E000120
c     compute the elapsed time                                          E000130
      if(ihnow.lt.ihbgn) ihnow=ihnow+24                                 E000140
      tsnow=3600.0*float(ihnow)+60.0*float(imnow)                       E000150
     +     +float(isnow)+float(ilnow)/100.0                             E000160
      timela=tsnow-tsbgn                                                E000170
c                                                                       E000180
c     estimate the remaining time                                       E000190
      timrem=(1.0/frcmp-1.0)*timela                                     E000200
c                                                                       E000210
c     convert the time units                                            E000220
      call ctime(timela,uniela)                                         E000230
      call ctime(timrem,unirem)                                         E000240
c                                                                       E000250
      return                                                            E000260
c                                                                       E000270
      end                                                               E000280
c                                                                       E000290
c*********************************************************************** E000300

c*********************************************************************** C000010
c                                                                       C000020
      subroutine ctime(time,units)                                      C000030
c                                                                       C000040
c     converts time units                                               C000050
c                                                                       C000060
c     type declaration                                                  C000070
      character*8 units                                                 C000080
c                                                                       C000090
      if(time.gt.60.0) then                                             C000100
        time=time/60.0                                                  C000110
        if(time.gt.60.0) then                                           C000120
          time=time/60.0                                                C000130
          units=' hours  '                                              C000140
        else                                                            C000150
          units=' minutes'                                              C000160
        endif                                                           C000170
      else                                                              C000180
        units=' seconds'                                                C000190
      endif                                                             C000200
c                                                                       C000210
      return                                                            C000220
c                                                                       C000230
      end                                                               C000240
c                                                                       C000250
c*********************************************************************** C000260
```

```
c*********************************************************************** R000010
c                                                                       R000020
      subroutine rpini(rndnxt, slcnst,asldeg,alttud,wspeed,wdrdeg,      R000030
     +                 wdrrad,jxrng,jrkmax)                             R000040
c                                                                       R000050
c      initializes replication loop                                     R000060
c                                                                       R000070
c      common blocks                                                    R000080
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   R000090
      common/error/ meror,irerr,iferr,merout                            R000100
      common/prmtr/ slcmin,slcmax,altmin,altmax,                        R000110
     +              wspmin,wspmax,wdrmin,wdrmax                         R000120
      common/rangs/ nrsegs,nrsegp                                       R000130
      common/pmiss/ pn(97),pnd(97),pnb(97),pnm(97)                      R000140
c                                                                       R000150
c      type declaration                                                 R000160
      double precision rndnxt                                           R000170
c                                                                       R000180
c      report call                                                      R000190
      if(merout.gt.0) write(2,'(2x,a)') 'rpini called'                  R000200
c                                                                       R000210
c      randomize parameters                                             R000220
      slcnst=rnval(slcmin,slcmax,rndnxt)                                R000230
      asldeg=10.8*slcnst**0.38                                          R000240
      alttud=rnval(altmin,altmax,rndnxt)                                R000250
      wspeed=rnval(wspmin,wspmax,rndnxt)                                R000260
      wdrdeg=rnval(wdrmin,wdrmax,rndnxt)                                R000270
      wdrrad=degrad*wdrdeg                                              R000280
c                                                                       R000290
c      set integration parameters                                       R000300
      if((wspeed.eq.0.0).or.(wdrdeg.eq.0.0)) then                       R000310
        jxrng=0                                                         R000320
        jrkmax=4                                                        R000330
      else                                                              R000340
        jxrng=1                                                         R000350
        jrkmax=6                                                        R000360
      endif                                                             R000370
c                                                                       R000380
c      initialize the miss probabilities                                R000390
      call arini(nrsegp,pn,pnd,pnb,pnm,1.0)                             R000400
c                                                                       R000410
c      report return                                                    R000420
      if(merout.gt.0) write(2,'(2x,a)') 'returning from rpini'          R000430
c                                                                       R000440
      return                                                            R000450
c                                                                       R000460
      end                                                               R000470
c                                                                       R000480
c*********************************************************************** R000490
```

```
c*********************************************************************** F000010
c                                                                       F000020
      subroutine frini(irep,ifrag,rndnxt, irico,lrange,htfrft,mfrht1,   F000030
     +                 vfrag,seldeg,selrad,aeldeg,aelrad,cd1,cd2,cd3,   F000040
     +                  deldst)                                         F000050
c                                                                       F000060
c      initializes fragment loop                                        F000070
c                                                                       F000080
c      common blocks                                                    F000090
       common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum  F000100
       common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2, F000110
     +               hcam3                                              F000120
       common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,fftmm,fmtft,fi2tc2      F000130
       common/error/ meror,irerr,iferr,merout                           F000140
       common/zones/ elzdeg,azsdeg,azsrad                               F000150
       common/donor/ nunits,htstft,htbaft                               F000160
       common/fragd/ famil(900),vfifs(900),farma(900),                  F000170
     +               fatn(900),ezldeg(900)                              F000180
       common/acptr/ htacft,wdacft,dpacft,htacfm                        F000190
       common/rkvar/ rkv(7)                                             F000200
c                                                                       F000210
c      type declaration                                                 F000220
       double precision rndnxt                                          F000230
c                                                                       F000240
c      set the error output flag                                        F000250
       merout=0                                                         F000260
       if(meror.gt.0) then                                              F000270
         if(irep.gt.irerr) then                                         F000280
           if(ifrag.gt.iferr) then                                      F000290
             merout=1                                                   F000300
           endif                                                        F000310
         endif                                                          F000320
       endif                                                            F000330
c                                                                       F000340
c      report call                                                      F000350
       if(merout.gt.0)                                                  F000360
     + write(2,'(2x,a,i3)') 'frini called at fragment ',ifrag           F000370
c                                                                       F000380
c      initialize the ricochet count and last range segment index       F000390
       irico=0                                                          F000400
       lrange=0                                                         F000410
c                                                                       F000420
c      randomize the initial fragment height                            F000430
       htfrft=rnval(htbaft,htstft,rndnxt)                               F000440
c                                                                       F000450
c      set switch 1 for fragment height                                 F000460
       if(htfrft.ge.htacft) then                                        F000470
         mfrht1=1                                                       F000480
       else                                                             F000490
         mfrht1=0                                                       F000500
       endif                                                            F000510
c                                                                       F000520
c      randomize the initial fragment velocity                          F000530
       rndlog=sqrt(-2.*log(rndom(rndnxt)))                              F000540
       rndnum=tpi*rndom(rndnxt)                                         F000550
       rndcos=rndlog*cos(rndnum)                                        F000560
       rndsin=rndlog*sin(rndnum)                                        F000570
       vfrag=vfifs(ifrag)+vfifs(ifrag)*0.035*rndcos                     F000580
c                                                                       F000590
c      randomize the initial fragment elevation angle                   F000600
       seldeg=(ezldeg(ifrag)+elzdeg*rndom(rndnxt))                      F000610
       if(seldeg.gt.89.99) then                                         F000620
         seldeg=89.99                                                   F000630
       else                                                             F000640
         if((seldeg.lt.1.0e-2).and.(seldeg.ge.0.0)) then                F000650
           seldeg=1.0e-2                                                F000660
         else                                                           F000670
           if((seldeg.lt.0.0).and.(seldeg.gt.-1.0e-2)) seldeg=-1.0e-2   F000680
         endif                                                          F000690
       endif                                                            F000700
       selrad=degrad*seldeg                                             F000710
       aeldeg=abs(seldeg)                                               F000720
```

```
      aelrad=abs(selrad)                                                F000730
c                                                                       F000740
c      randomize drag coefficient parameters                            F000750
      cdmax=1.75*farma(ifrag)-1.27                                      F000760
      cdmin=0.66*farma(ifrag)-0.26                                      F000770
      cd0=rnval(cdmin,cdmax,rndnxt)                                     F000780
      cd1=cd0+0.2                                                       F000790
      cd2=cd0+0.65                                                      F000800
      cd3=cd0+0.5                                                       F000810
c                                                                       F000820
c      initialize the integration step                                  F000830
      call dstep(htfrft,seldeg,selrad, deldst)                          F000840
c                                                                       F000850
c      initialize the integration variables                             F000860
      rkv(1)=0.0                                                        F000870
      rkv(2)=vfrag*cos(selrad)                                          F000880
      rkv(3)=0.0                                                        F000890
      rkv(4)=vfrag*sin(selrad)                                          F000900
      rkv(5)=htfrft                                                     F000910
      rkv(6)=0.0                                                        F000920
      rkv(7)=0.0                                                        F000930
c                                                                       F000940
c      report return                                                    F000950
      if(merout.gt.0) write(2,'(2x,a)') 'returning from frini'          F000960
c                                                                       F000970
      return                                                            F000980
c                                                                       F000990
      end                                                               F001000
c                                                                       F001010
c*********************************************************************** F001020

c*********************************************************************** D000010
c                                                                       D000020
      subroutine dstep(htfrft,seldeg,selrad, deldst)                    D000030
c                                                                       D000040
c      determiness integration distance step                            D000050
c                                                                       D000060
c      common blocks                                                    D000070
      common/error/ meror,irerr,iferr,merout                            D000080
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           D000090
c                                                                       D000100
c      report call                                                      D000110
      if(merout.gt.0) write(2,'(4x,a)') 'dstep called'                  D000120
c                                                                       D000130
      if(seldeg.lt.0.0) then                                            D000140
        dtognd=htfrft/sin(-selrad)                                      D000150
        if(dtognd.gt.rngsix) then                                       D000160
          deldst=rngsgh                                                 D000170
        else                                                            D000180
          deldst=aint(dtognd/1.2)                                       D000190
        endif                                                           D000200
      else                                                              D000210
        if(seldeg.gt.70.0) then                                         D000220
          deldst=rngsgh                                                 D000230
        else                                                            D000240
          deldst=rngscl*aint((-0.02492*seldeg+2.20134)*seldeg+18.8306)  D000250
        endif                                                           D000260
      endif                                                             D000270
c                                                                       D000280
c      report return                                                    D000290
      if(merout.gt.0) write(2,'(4x,a)') 'returning from dstep'          D000300
c                                                                       D000310
      return                                                            D000320
c                                                                       D000330
      end                                                               D000340
c                                                                       D000350
c*********************************************************************** D000360
```

```
c*********************************************************************** T000010
c                                                                       T000020
      subroutine tstep(deldst,irico,vfrag,aelrad, deltim)               T000030
c                                                                       T000040
c      determiness integration time step                                T000050
c                                                                       T000060
c      common blocks                                                    T000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   T000080
      common/error/ meror,irerr,iferr,merout                            T000090
      common/acptr/ htacft,wdacft,dpacft,htacfm                         T000100
      common/rkvar/ rkv(7)                                              T000110
c                                                                       T000120
c      report call                                                      T000130
      if(merout.gt.0) write(2,'(2x,a)') 'tstep called'                  T000140
c                                                                       T000150
c      compute the standard time step                                   T000160
      deltim=deldst/vfrag                                               T000170
c                                                                       T000180
c      compute a special time step if approaching                       T000190
c      the top of thr hazard volume or the ground                       T000200
      if(rkv(4).lt.0.0) then                                            T000210
c                                                                       T000220
        elerad=max(aelrad,smlrad)                                       T000230
        deldsp=deldst+htacfm                                            T000240
c                                                                       T000250
c        if the fragment is below the top of the hazard volume          T000260
        if(rkv(5).le.htacft) then                                       T000270
c                                                                       T000280
c          compute a time step approaching the ground                   T000290
          dtognd=rkv(5)/sin(elerad)                                     T000300
          if(dtognd.lt.deldsp) deltim=(dtognd-htacfm)/vfrag             T000310
          if(dtognd.lt.htacft) deltim=(dtognd-0.3)/vfrag                T000320
          if(dtognd.lt.0.4   ) deltim=dtognd/vfrag+2.0e-6               T000330
c                                                                       T000340
        else                                                            T000350
c                                                                       T000360
c          if the fragment is low enough                                T000370
          if(rkv(5).lt.(deldst+12.0)) then                              T000380
c                                                                       T000390
c            compute a time step approaching                            T000400
c            the top of thr hazard volume                               T000410
            dtotop=(rkv(5)-htacft)/sin(elerad)                          T000420
            if(dtotop.lt.deldsp) deltim=(dtotop-htacfm)/vfrag           T000430
            if(dtotop.lt.htacft) deltim=(dtotop-0.3)/vfrag              T000440
            if(dtotop.lt.0.4   ) deltim=dtotop/vfrag+2.0e-6             T000450
c                                                                       T000460
          endif                                                         T000470
c                                                                       T000480
        endif                                                           T000490
c                                                                       T000500
      endif                                                             T000510
c                                                                       T000520
c      limit the time step                                              T000530
      if((vfrag.lt.100.0).and.(rkv(4).gt.-30.0).and.(deltim.gt.0.1))    T000540
     + deltim=0.1                                                       T000550
      if((irico.gt.1).and.(vfrag.lt.150.0).and.(deltim.gt.0.08))        T000560
     + deltim=0.08                                                      T000570
c                                                                       T000580
c      report return                                                    T000590
      if(merout.gt.0) write(2,'(2x,a)') 'returning from tstep'          T000600
c                                                                       T000610
      return                                                            T000620
c                                                                       T000630
      end                                                               T000640
c                                                                       T000650
c*********************************************************************** T000660
```

```
c*********************************************************************** R000010
c                                                                       R000020
      subroutine rkint(wspeed,wdrdeg,wdrrad,jxrng,jrkmax,cd1,cd2,cd3,   R000030
     +                 deltim,alttud,ifrag)                             R000040
c                                                                       R000050
c      performs Runge-Kutta integration                                 R000060
c                                                                       R000070
c      common blocks                                                    R000080
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   R000090
      common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2,  R000100
     +              hcam3                                              R000110
      common/error/ meror,irerr,iferr,merout                            R000120
      common/fragd/ famil(900),vfifs(900),farma(900),                   R000130
     +              fatn(900),ezldeg(900)                               R000140
      common/rkvar/ rkv(7)                                              R000150
c                                                                       R000160
c      local arrays                                                     R000170
      dimension rko(7),rks(7),rk(4,7)                                   R000180
c                                                                       R000190
c      report call                                                      R000200
      if(merout.gt.0) write(2,'(2x,a)') 'rkint called'                  R000210
c                                                                       R000220
c      begin Runge-Kutta outer loop                                     R000230
      do 200 irk=1,4                                                    R000240
c                                                                       R000250
      if(irk.lt.3) then                                                 R000260
        rkcnst=0.5                                                      R000270
      else                                                              R000280
        rkcnst=1.                                                       R000290
      endif                                                             R000300
c                                                                       R000310
c      determine relative wind                                          R000320
      if(wspeed.le.0.0) then                                            R000330
        vfrel=sqrt(rkv(2)*rkv(2)+rkv(4)*rkv(4))                         R000340
        aayrad=atan(rkv(4)/rkv(2))                                      R000350
        aaxrad=hpi-aayrad                                               R000360
      else                                                              R000370
        if(wdrdeg.eq.0.0) then                                          R000380
          vxfrel=rkv(2)-wspeed                                          R000390
          vyfrel=rkv(4)                                                 R000400
          vfrel=sqrt(vxfrel*vxfrel+vyfrel*vyfrel)                       R000410
          aayrad=atan(vyfrel/abs(vxfrel))                               R000420
          aaxrad=atan(vxfrel/abs(vyfrel))                               R000430
        else                                                            R000440
          vxfrel=rkv(2)-wspeed*cos(wdrrad)                              R000450
          vyfrel=rkv(4)                                                 R000460
          vzfrel=wspeed*sin(wdrrad)-rkv(6)                              R000470
          vfrel=sqrt(vxfrel*vxfrel+vyfrel*vyfrel+vzfrel*vzfrel)         R000480
          aayrad=atan(vyfrel/sqrt(vxfrel*vxfrel+vzfrel*vzfrel))         R000490
          aaxrad=atan(vxfrel/sqrt(vyfrel*vyfrel+vzfrel*vzfrel))         R000500
          aazrad=atan(vzfrel/sqrt(vxfrel*vxfrel+vyfrel*vyfrel))         R000510
        endif                                                           R000520
      endif                                                             R000530
c                                                                       R000540
```

```
c      determine air density, Mach number and drag coefficient          R000550
       hdelt=-((rkv(5)+alttud)+rkv(4)*0.5*deltim)                       R000560
       hrhair=hrair0*expon(hdelt/racnst)                                R000570
       hcam=vfrel/(vsound*expon(hdelt/hccnst))                          R000580
       if(hcam.ge.hcam1) then                                           R000590
         if(hcam.ge.hcam2) then                                         R000600
           if(hcam.ge.hcam3) then                                       R000610
             cd=cd3                                                     R000620
           else                                                         R000630
             cd=cd2-0.15/(hcam3-hcam2)*(hcam-hcam2)                     R000640
           endif                                                        R000650
         else                                                           R000660
           cd=cd2-0.45/(hcam2-hcam1)*(hcam2-hcam)                       R000670
         endif                                                          R000680
       else                                                             R000690
         cd=cd1-0.2/(hcam1-0.1)*(hcam1-hcam)                            R000700
       endif                                                            R000710
c                                                                       R000720
c      compute velocities and accelerations                             R000730
       acdrag=hrhair*vfrel*vfrel*cd*famil(ifrag)/144.                   R000740
       rko(2)=-acdrag*sin(aaxrad)                                       R000750
       rko(3)=rkv(2)                                                    R000760
       rko(4)=-acdrag*sin(aayrad)-gg                                    R000770
       rko(5)=rkv(4)                                                    R000780
       if(jxrng.ne.0) then                                              R000790
         rko(6)=acdrag*sin(aazrad)                                      R000800
         rko(7)=rkv(6)                                                  R000810
       endif                                                            R000820
c                                                                       R000830
c      begin Runge-Kutta inner loop                                     R000840
       do 100 jrk=2,jrkmax,2                                            R000850
         if(.not.(irk.gt.1)) rks(jrk)=rkv(jrk)                          R000860
         rk(irk,jrk)=rko(jrk)*deltim                                    R000870
         if(irk.eq.4) then                                              R000880
           rkv(jrk)=rks(jrk)+                                           R000890
     +      (rk(1,jrk)+2*rk(2,jrk)+2*rk(3,jrk)+rk(4,jrk))/6.0           R000900
           rkv(jrk+1)=rkv(jrk+1)+                                       R000910
     +     deltim*(rks(jrk)+(rk(1,jrk)+rk(2,jrk)+rk(3,jrk))/6.0)        R000920
         else                                                           R000930
           rkv(jrk)=rks(jrk)+rk(irk,jrk)*rkcnst                         R000940
         endif                                                          R000950
  100  continue                                                         R000960
c                                                                       R000970
  200  continue                                                         R000980
c                                                                       R000990
c      report return                                                    R001000
       if(merout.gt.0) write(2,'(2x,a)') 'returning from rkint'         R001010
c                                                                       R001020
       return                                                           R001030
c                                                                       R001040
       end                                                              R001050
c                                                                       R001060
c*********************************************************************** R001070
```

```
c*********************************************************************** F000010
c                                                                       F000020
      subroutine frvar(vfrag,selrad,seldeg,aelrad,aeldeg,range,irange)  F000030
c                                                                       F000040
c      determines the fragment variable values                          F000050
c                                                                       F000060
c      common blocks                                                    F000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   F000080
      common/error/ meror,irerr,iferr,merout                           F000090
      common/rangs/ nrsegs,nrsegp                                      F000100
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin          F000110
      common/rkvar/ rkv(7)                                             F000120
c                                                                       F000130
c      report call                                                      F000140
      if(merout.gt.0) write(2,'(2x,a)') 'frvar called'                 F000150
c                                                                       F000160
c      compute the fragment velocity                                    F000170
      vfrag=sqrt(rkv(2)*rkv(2)+rkv(4)*rkv(4)+rkv(6)*rkv(6))            F000180
c                                                                       F000190
c      compute the elevation angle                                      F000200
      selrad=atan(rkv(4)/sqrt(rkv(2)*rkv(2)+rkv(6)*rkv(6)))            F000210
      seldeg=raddeg*selrad                                             F000220
      aelrad=abs(selrad)                                               F000230
      aeldeg=abs(seldeg)                                               F000240
c                                                                       F000250
c      compute the range and segment index                              F000260
      range=sqrt(rkv(3)*rkv(3)+rkv(7)*rkv(7))                          F000270
      irange=int(range/rngseg)+1                                       F000280
      call irlim(irange,nrsegs,nrsegp)                                 F000290
c                                                                       F000300
c      report return                                                    F000310
      if(merout.gt.0) write(2,'(2x,a)') 'returning from frvar'         F000320
c                                                                       F000330
      return                                                           F000340
c                                                                       F000350
      end                                                              F000360
c                                                                       F000370
c*********************************************************************** F000380

c*********************************************************************** I000010
c                                                                       I000020
      subroutine irlim(irange,nrsegs,nrsegp)                           I000030
c                                                                       I000040
c      limits the value of irange                                       I000050
c                                                                       I000060
      if(irange.lt.1) irange=1                                         I000070
      if(irange.gt.nrsegs) irange=nrsegp                               I000080
c                                                                       I000090
      return                                                           I000100
c                                                                       I000110
      end                                                              I000120
c                                                                       I000130
c*********************************************************************** I000140
```

```
c*********************************************************************** H000010
c                                                                       H000020
      subroutine hazch(ifrag,irange,mfrht1,mfrht2,lrange,aelrad,vfrag)  H000030
c                                                                       H000040
c      directs hazard computations if they are required                 H000050
c                                                                       H000060
c      common blocks                                                    H000070
      common/error/ meror,irerr,iferr,merout                            H000080
      common/rangs/ nrsegs,nrsegp                                       H000090
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           H000100
      common/acptr/ htacft,wdacft,dpacft,htacfm                         H000110
      common/rkvar/ rkv(7)                                              H000120
c                                                                       H000130
c      report call                                                      H000140
      if(merout.gt.0) write(2,'(2x,a)') 'hazch called'                  H000150
c                                                                       H000160
c      set switch 2 for fragment height                                 H000170
      if(rkv(5).gt.htacft) then                                         H000180
        mfrht2=1                                                        H000190
      else                                                              H000200
        mfrht2=0                                                        H000210
      endif                                                             H000220
c                                                                       H000230
      if(mfrht2.eq.1) then                                              H000240
        if(mfrht1.ne.1) then                                            H000250
          mfrht1=1                                                      H000260
          delrng=(rkv(5)-htacft)/tan(aelrad)                            H000270
          irange=                                                       H000280
     +    int((sqrt(rkv(3)*rkv(3)+rkv(7)*rkv(7))-delrng)/rngseg)+1      H000290
          call irlim(irange,nrsegs,nrsegp)                              H000300
          call hazrd(ifrag,irange,lrange,aelrad,vfrag)                  H000310
        endif                                                           H000320
      else                                                              H000330
        call hazrd(ifrag,irange,lrange,aelrad,vfrag)                    H000340
      endif                                                             H000350
c                                                                       H000360
c      report return                                                    H000370
      if(merout.gt.0) write(2,'(2x,a)') 'returning from hazch'          H000380
c                                                                       H000390
      return                                                            H000400
c                                                                       H000410
      end                                                               H000420
c                                                                       H000430
c*********************************************************************** H000440
```

```
c*********************************************************************** H000010
c                                                                       H000020
      subroutine hazrd(ifrag,irange,lrange,aelrad,vfrag)                H000030
c                                                                       H000040
c      performs hazard computations                                     H000050
c                                                                       H000060
c      common blocks                                                    H000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   H000080
      common/error/ meror,irerr,iferr,merout                            H000090
      common/zones/ elzdeg,azsdeg,azsrad                                H000100
      common/fragd/ famil(900),vfifs(900),farma(900),                   H000110
     +              fatn(900),ezldeg(900)                               H000120
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           H000130
      common/arang/ rangem(97),ahfrn(97),ahtop(97)                      H000140
      common/acptr/ htacft,wdacft,dpacft,htacfm                         H000150
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       H000160
      common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw                       H000170
      common/thorr/ athr,bthr,cthr,dthr,vrrms,tcr                       H000180
      common/thras/ athasv,bthasv,cthasv,tcasv,                         H000190
     +              athasm,bthasm,cthasm,tcasm                          H000200
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 H000210
      common/rkvar/ rkv(7)                                              H000220
      common/pmiss/ pn(97),pnd(97),pnb(97),pnm(97)                      H000230
      common/denst/ fd(97),fdd(97),fdb(97),fdm(97)                      H000240
      common/phitt/ ph(97),phd(97),phb(97),phm(97)                      H000250
c                                                                       H000260
c      report call                                                      H000270
      if(merout.gt.0) write(2,'(4x,a)') 'hazrd called'                  H000280
c                                                                       H000290
c      initialize parameters if range segment index has changed         H000300
      if(irange.ne.lrange) then                                         H000310
        call wiini(itrpt,pn0,pn0d,pn0b,pn0m,pnr,pnrd,pnrb,pnrm,         H000320
     +                  fd0,fd0d,fd0b,fd0m,fdr,fdrd,fdrb,fdrm)          H000330
      endif                                                             H000340
c                                                                       H000350
c      increment within segment counter                                 H000360
      itrpm=itrpt                                                       H000370
      itrpt=itrpt+1                                                     H000380
c                                                                       H000390
c      compute the sine and cosine of the elevation angle               H000400
      sinele=sin(aelrad)                                                H000410
      cosele=cos(aelrad)                                                H000420
c                                                                       H000430
c      compute the hazard volume presented area                         H000440
      apvft2=ahfrn(irange)*cosele+ahtop(irange)*sinele                  H000450
c                                                                       H000460
c      compute the total presented area of the acceptor                 H000470
      apaft2=htacft*cosele                                              H000480
      if(rkv(4).lt.0.0) apaft2=apaft2+dpacft*sinele                     H000490
      apaft2=apaft2*wdacft                                              H000500
c                                                                       H000510
c      check and adjust fragment parameters for shroud penetration      H000520
      call shpen(ifrag,vfrag, vfhaz,vfcdx,vfcdp,vfcbx,vfcbp)           H000530
c                                                                       H000540
c      accumulate the unconditional miss probability                    H000550
      call nohit(ifrag,apaft2,apvft2,irange,itrpm,itrpt,pnr,pn0,pn)     H000560
c                                                                       H000570
c      accumulate the unconditional fragment density                    H000580
      call dnsty(ifrag,apvft2,irange,itrpm,itrpt,fdr,fd0,fd)            H000590
```

```
c                                                                       H000600
c      determine the detonation vulnerable area                         H000610
       avdft2=0.0                                                       H000620
       if(macwh.ne.0) then                                              H000630
         call vulnd(vfcdx,vfhaz,axfrn,axtop,cosele,sinele, avdft2)      H000640
       endif                                                            H000650
       if(macrm.ne.0) then                                              H000660
         call vulnd(vfcdp,vfhaz,apfrn,aptop,cosele,sinele, avdft2)      H000670
       endif                                                            H000680
c                                                                       H000690
c      accumulate the detonation miss probability and fragment density  H000700
       call nohit(ifrag,avdft2,apvft2,irange,itrpm,itrpt,pnrd,pn0d,pnd) H000710
       if(avdft2.ne.0.0) then                                           H000720
         call dnsty(ifrag,apvft2,irange,itrpm,itrpt,fdrd,fd0d,fdd)      H000730
       endif                                                            H000740
c                                                                       H000750
c      determine the burn vulnerable area                               H000760
       avbft2=0.0                                                       H000770
       if(macwh.ne.0) then                                              H000780
         call vulnb(vfcbx,cthw,dthw,vrwms,vfhaz,axfrn,axtop,cosele,     H000790
     +             sinele, avbft2)                                      H000800
       endif                                                            H000810
       if(macrm.ne.0) then                                              H000820
         call vulnb(vfcbp,cthr,dthr,vrrms,vfhaz,apfrn,aptop,cosele,     H000830
     +             sinele, avbft2)                                      H000840
       endif                                                            H000850
c                                                                       H000860
c      accumulate the burn miss probability and fragment density        H000870
       call nohit(ifrag,avbft2,apvft2,irange,itrpm,itrpt,pnrb,pn0b,pnb) H000880
       if(avbft2.ne.0.0) then                                           H000890
         call dnsty(ifrag,apvft2,irange,itrpm,itrpt,fdrb,fd0b,fdb)      H000900
       endif                                                            H000910
c                                                                       H000920
c      determine the mechanical damage vulnerable area                  H000930
       call vulnm(ifrag,vfhaz,cosele,sinele, avmft2)                    H000940
c                                                                       H000950
c      accumulate the mechanical damage                                 H000960
c      miss probability and fragment density                            H000970
       call nohit(ifrag,avmft2,apvft2,irange,itrpm,itrpt,pnrm,pn0m,pnm) H000980
       if(avmft2.ne.0.0) then                                           H000990
         call dnsty(ifrag,apvft2,irange,itrpm,itrpt,fdrm,fd0m,fdm)      H001000
       endif                                                            H001010
c                                                                       H001020
       lrange=irange                                                    H001030
c                                                                       H001040
c      report return                                                    H001050
       if(merout.gt.0) write(2,'(4x,a)') 'returning from hazrd'         H001060
c                                                                       H001070
       return                                                           H001080
c                                                                       H001090
       end                                                              H001100
c                                                                       H001110
c*********************************************************************** H001120
```

```
c*********************************************************************** W000010
c                                                                       W000020
      subroutine wiini(itrpt,pn0,pn0d,pn0b,pn0m,pnr,pnrd,pnrb,pnrm,     W000030
     +                       fd0,fd0d,fd0b,fd0m,fdr,fdrd,fdrb,fdrm)     W000040
c                                                                       W000050
c      initializes within segment parameters                            W000060
c                                                                       W000070
c      common block                                                     W000080
      common/error/ meror,irerr,iferr,merout                            W000090
c                                                                       W000100
c      report call                                                      W000110
      if(merout.gt.0) write(2,'(6x,a)') 'wiini called'                  W000120
c                                                                       W000130
c      initialize the within segment point index                        W000140
      itrpt=0                                                           W000150
c                                                                       W000160
c      initialize the miss probability parameters                       W000170
      pn0=1.0                                                           W000180
      pnr=0.0                                                           W000190
      pn0d=1.0                                                          W000200
      pnrd=0.0                                                          W000210
      pn0b=1.0                                                          W000220
      pnrb=0.0                                                          W000230
      pn0m=1.0                                                          W000240
      pnrm=0.0                                                          W000250
c                                                                       W000260
c      initialize the fragment density parameters                       W000270
      fdr=0.0                                                           W000280
      fd0=0.0                                                           W000290
      fdrd=0.0                                                          W000300
      fd0d=0.0                                                          W000310
      fdrb=0.0                                                          W000320
      fd0b=0.0                                                          W000330
      fdrm=0.0                                                          W000340
      fd0m=0.0                                                          W000350
c                                                                       W000360
c      report return                                                    W000370
      if(merout.gt.0) write(2,'(6x,a)') 'returning from wiini'          W000380
c                                                                       W000390
      return                                                            W000400
c                                                                       W000410
      end                                                               W000420
c                                                                       W000430
c*********************************************************************** W000440
```

```
c*********************************************************************** S000010
c                                                                       S000020
      subroutine shpen(ifrag,vfrag, vfhaz,vfcdx,vfcdp,vfcbx,vfcbp)      S000030
c                                                                       S000040
c     checks and adjusts fragment parameters for shroud penetration     S000050
c                                                                       S000060
c      common blocks                                                    S000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   S000080
      common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmtft,fi2tc2      S000090
      common/error/ meror,irerr,iferr,merout                            S000100
      common/fragd/ famil(900),vfifs(900),farma(900),                   S000110
     +              fatn(900),ezldeg(900)                               S000120
      common/fragc/ facm2(900),fdmm(900),fmgm(900)                      S000130
      common/fragx/ vcrdx(900),vcrbx(900)                               S000140
      common/fragp/ vcrdp(900),vcrbp(900)                               S000150
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       S000160
      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin                 S000170
      common/jrdtx/ abjrx,cjrx                                          S000180
      common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw                       S000190
      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin                 S000200
      common/jrdtp/ abjrp,cjrp                                          S000210
      common/thorr/ athr,bthr,cthr,dthr,vrrms,tcr                       S000220
      common/geoas/ ashcm                                               S000230
      common/thras/ athasv,bthasv,cthasv,tcasv,                         S000240
     +              athasm,bthasm,cthasm,tcasm                          S000250
c                                                                       S000260
c      report call                                                      S000270
      if(merout.gt.0) write(2,'(6x,a)') 'shpen called'                  S000280
c                                                                       S000290
      if(macsh.eq.0) then                                               S000300
c                                                                       S000310
c        no shroud is present                                           S000320
c                                                                       S000330
c        assign unadjusted values to hazard fragment parameters         S000340
         vfhaz=vfrag                                                    S000350
         vfcdx=vcrdx(ifrag)                                             S000360
         vfcdp=vcrdp(ifrag)                                             S000370
         vfcbx=vcrbx(ifrag)                                             S000380
         vfcbp=vcrbp(ifrag)                                             S000390
c                                                                       S000400
      else                                                              S000410
c                                                                       S000420
c        a shroud is present                                            S000430
c                                                                       S000440
c        adjust the fragment velocity and mass for shroud penetration   S000450
         call vmthr(ashcm,facm2(ifrag),vfrag,fmgm(ifrag),tcasv,athasv,  S000460
     +              bthasv,cthasv,tcasm,athasm,bthasm,cthasm, vfhaz,    S000470
     +               fmhaz)                                             S000480
c                                                                       S000490
c        compute the hazard fragment area in sq cm                      S000500
         fhacm2=fi2tc2*famil(ifrag)*fgmtlb*fmhaz                        S000510
c                                                                       S000520
c        compute the hazard fragment diameter in mm                     S000530
         fhdmm=2.0*sqrt(100.0*fhacm2/pi)                                S000540
c                                                                       S000550
c        compute the explosive detonation critical velocity             S000560
         vfcdx=vcrjr(abjrx,cjrx,whhmm,fhdmm)                            S000570
c                                                                       S000580
c        compute the propellant detonation critical velocity            S000590
         vfcdp=vcrjr(abjrp,cjrp,rmhmm,fhdmm)                            S000600
c                                                                       S000610
c         compute the explosive burning critical velocity               S000620
         vfcbx=vcrth(tcw,athw,bthw,whhcm,fhacm2,fmhaz)                  S000630
c                                                                       S000640
c        compute the propellant burning critical velocity               S000650
         vfcbp=vcrth(tcr,athr,bthr,rmhcm,fhacm2,fmhaz)                  S000660
c                                                                       S000670
      endif                                                             S000680
c                                                                       S000690
c      report return                                                    S000700
      if(merout.gt.0) write(2,'(6x,a)') 'returning from shpen'          S000710
c                                                                       S000720
      return                                                            S000730
c                                                                       S000740
      end                                                               S000750
c                                                                       S000760
c*********************************************************************** S000770
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vmthr(shhcm,facm2,vffs,fmgm,tcsv,athsv,bthsv,cthsv,    V000030
     +                 tcsm,athsm,bthsm,cthsm, vf,fm)                   V000040
c                                                                       V000050
c      adjusts the fragment velocity and mass for shroud penetration    V000060
c                                                                       V000070
c      common blocks                                                    V000080
       common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,fftmm,fmtft,fi2tc2      V000090
       common/error/ meror,irerr,iferr,merout                           V000100
c                                                                       V000110
c      report call                                                      V000120
       if(merout.gt.0) write(2,'(8x,a)') 'vmthr called'                 V000130
c                                                                       V000140
c      compute the plug volume in sq cm                                 V000150
       ha=shhcm*facm2                                                   V000160
c                                                                       V000170
c      convert the fragment velocity from ft/s to m/s                   V000180
       vfms=ffttm*vffs                                                  V000190
c                                                                       V000200
c      adjust the fragment velocity in m/s using THOR (zero-obliquity)  V000210
       vf=vfms-tcsv*(ha**athsv)*(fmgm**bthsv)*(vfms**cthsv)             V000220
       if(vf.lt.0.0) vf=0.0                                             V000230
c                                                                       V000240
c      adjust the fragment mass in gm using THOR (zero-obliquity)       V000250
       fm=fmgm-tcsm*(ha**athsm)*(fmgm**bthsm)*(vfms**cthsm)             V000260
       if(fm.lt.0.0) fm=0.0                                             V000270
c                                                                       V000280
c      convert the hazard fragment velocity from m/s to ft/s            V000290
       vf=fmtft*vf                                                      V000300
c                                                                       V000310
c      report return                                                    V000320
       if(merout.gt.0) write(2,'(8x,a)') 'returning from vmthr'         V000330
c                                                                       V000340
       return                                                           V000350
c                                                                       V000360
       end                                                              V000370
c                                                                       V000380
c*********************************************************************** V000390

c*********************************************************************** V000010
c                                                                       V000020
       subroutine vulnd(vcrd,vfhaz,afrn,atop,cosele,sinele, avdft2)     V000030
c                                                                       V000040
c      determines the detonation vulnerable area                        V000050
c                                                                       V000060
c      common blocks                                                    V000070
       common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,fftmm,fmtft,fi2tc2      V000080
       common/error/ meror,irerr,iferr,merout                           V000090
c                                                                       V000100
c      report call                                                      V000110
       if(merout.gt.0) write(2,'(6x,a)') 'vulnd called'                 V000120
c                                                                       V000130
c      convert the fragment velocity from ft/s to mm/us                 V000140
       vfmmus=fftmm*vfhaz                                               V000150
c                                                                       V000160
c      check the detonation condition                                   V000170
       if(vfmmus.gt.vcrd) then                                          V000180
c                                                                       V000190
c         the fragment can produce detonation                           V000200
c                                                                       V000210
c         compute the maximum obliquity                                 V000220
          cosobx=vcrd/vfmmus                                            V000230
c                                                                       V000240
c         compute the vulnerable area for initiation of detonation      V000250
          call iniar(cosobx,cosele,sinele,afrn,atop, avdft2)            V000260
c                                                                       V000270
       endif                                                            V000280
c                                                                       V000290
c      report return                                                    V000300
       if(merout.gt.0) write(2,'(6x,a)') 'returning from vulnd'         V000310
c                                                                       V000320
       return                                                           V000330
c                                                                       V000340
       end                                                              V000350
c                                                                       V000360
c*********************************************************************** V000370
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulnb(vcrb,cth,dth,vrms,vfhaz,afrn,atop,cosele,sinele, V000030
     +                 avbft2)                                          V000040
c                                                                       V000050
c      determines the burn vulnerable area                              V000060
c                                                                       V000070
c      common blocks                                                    V000080
      common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmtft,fi2tc2      V000090
      common/error/ meror,irerr,iferr,merout                           V000100
c                                                                       V000110
c      report call                                                      V000120
      if(merout.gt.0) write(2,'(6x,a)') 'vulnb called'                 V000130
c                                                                       V000140
c      convert the fragment velocity from ft/s to m/s                   V000150
      vfms=ffttm*vfhaz                                                 V000160
c                                                                       V000170
c      compute the burn critical condition using THOR                   V000180
      vright=vcrb*vfms**cth                                            V000190
      vleft=vfms-vrms                                                  V000200
c                                                                       V000210
c      check the burn condition                                         V000220
      if(vleft.gt.vright) then                                         V000230
c                                                                       V000240
c        the fragment can produce burning                               V000250
c                                                                       V000260
c        compute the maximum obliquity                                  V000270
        cosobx=(vright/vleft)**dth                                     V000280
c                                                                       V000290
c        compute the vulnerable area for initiation of burning          V000300
        call iniar(cosobx,cosele,sinele,afrn,atop, avbft2)             V000310
c                                                                       V000320
      endif                                                            V000330
c                                                                       V000340
c      report return                                                    V000350
      if(merout.gt.0) write(2,'(6x,a)') 'returning from vulnb'         V000360
c                                                                       V000370
      return                                                           V000380
c                                                                       V000390
      end                                                              V000400
c                                                                       V000410
c*********************************************************************** V000420
```

```
c*********************************************************************** I000010
c                                                                       I000020
      subroutine iniar(cosobx,cosele,sinele,afrn,atop, avft2)           I000030
c                                                                       I000040
c      computes the vulnerable area for initiation                      I000050
c                                                                       I000060
c      common blocks                                                    I000070
      common/error/ meror,irerr,iferr,merout                            I000080
      common/acflg/ nweps,macwh,macrm,macsh,mstor                       I000090
      common/rkvar/ rkv(7)                                              I000100
c                                                                       I000110
c      report call                                                      I000120
      if(merout.gt.0) write(2,'(8x,a)') 'iniar called'                  I000130
c                                                                       I000140
      if(mstor.eq.1) then                                               I000150
c                                                                       I000160
c        the storage arrangement is vertical                            I000170
c                                                                       I000180
c        compute the maximum tangency                                   I000190
        costax=cosobx/cosele                                            I000200
c                                                                       I000210
c        if the maximum obliquity exceeds the elevation angle           I000220
        if(costax.lt.1.0) then                                          I000230
c                                                                       I000240
c          compute the front vulnerable area                            I000250
          avft2=avft2+cosele*sqrt(1.0-costax*costax)*afrn               I000260
c                                                                       I000270
        endif                                                           I000280
c                                                                       I000290
c        if the vertical fragment velocity component is negative        I000300
        if(rkv(4).lt.0.0) then                                          I000310
c                                                                       I000320
c          and if the elevation angle exceeds the                       I000330
c          complement of the maximum obliquity                          I000340
          if(sinele.gt.cosobx) then                                     I000350
c                                                                       I000360
c            compute the top vulnerable area                            I000370
            avft2=avft2+sinele*atop                                     I000380
c                                                                       I000390
          endif                                                         I000400
c                                                                       I000410
        endif                                                           I000420
c                                                                       I000430
      else                                                              I000440
c                                                                       I000450
c        the storage arrangement is horizontal                          I000460
c                                                                       I000470
c        compute the sine of the maximum obliquity                      I000480
        sinobx=sqrt(1.0-cosobx*cosobx)                                  I000490
c                                                                       I000500
c        compute the front vulnerable area                              I000510
        avft2=avft2+sinobx*afrn                                         I000520
c                                                                       I000530
c        if the vertical fragment velocity component is negative        I000540
        if(rkv(4).lt.0.0) then                                          I000550
c                                                                       I000560
c          compute the top vulnerable area                              I000570
          avft2=avft2+sinobx*atop                                       I000580
c                                                                       I000590
        endif                                                           I000600
c                                                                       I000610
      endif                                                             I000620
c                                                                       I000630
c      report return                                                    I000640
      if(merout.gt.0) write(2,'(8x,a)') 'returning from iniar'          I000650
c                                                                       I000660
      return                                                            I000670
c                                                                       I000680
      end                                                               I000690
c                                                                       I000700
c*********************************************************************** I000710
```

```
c*********************************************************************** V000010
c                                                                       V000020
      subroutine vulnm(ifrag,vfhaz,cosele,sinele, avmft2)               V000030
c                                                                       V000040
c      determines mechanical damage vulnerable area                     V000050
c                                                                       V000060
c      common blocks                                                    V000070
      common/error/ meror,irerr,iferr,merout                            V000080
      common/fragk/ frkec(900)                                          V000090
      common/mechd/ wpke                                                V000100
      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop                 V000110
      common/rkvar/ rkv(7)                                              V000120
c                                                                       V000130
c      report call                                                      V000140
      if(merout.gt.0) write(2,'(6x,a)') 'vulnm called'                  V000150
c                                                                       V000160
c      compute the kinetic energy                                       V000170
      frke=frkec(ifrag)*vfhaz*vfhaz                                     V000180
c                                                                       V000190
c      check the mechanical damage condition                            V000200
      if(wpke.lt.frke) then                                             V000210
c                                                                       V000220
c        the fragment produces mechanical damage                        V000230
        avmft2=amfrn*cosele                                             V000240
        if(rkv(4).lt.0.0) avmft2=avmft2+amtop*sinele                    V000250
c                                                                       V000260
      else                                                              V000270
c                                                                       V000280
c        the fragment does not produces mechanical damage               V000290
        avmft2=0.0                                                      V000300
c                                                                       V000310
      endif                                                             V000320
c                                                                       V000330
c      report return                                                    V000340
      if(merout.gt.0) write(2,'(6x,a)') 'returning from vulnm'          V000350
c                                                                       V000360
      return                                                            V000370
c                                                                       V000380
      end                                                               V000390
c                                                                       V000400
c*********************************************************************** V000410

c*********************************************************************** N000010
c                                                                       N000020
      subroutine nohit(ifrag,aptft2,apvft2,irange,itrpm,itrpt,pnrt,pn0t, N000030
     +                 pnt)                                             N000040
c                                                                       N000050
c      accumulates miss probability                                     N000060
c                                                                       N000070
c      common blocks                                                    N000080
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   N000090
      common/error/ meror,irerr,iferr,merout                            N000100
      common/fragd/ famil(900),vfifs(900),farma(900),                   N000110
     +              fatn(900),ezldeg(900)                               N000120
c                                                                       N000130
c      local array                                                      N000140
      dimension pnt(97)                                                 N000150
c                                                                       N000160
c      report call                                                      N000170
      if(merout.gt.0) write(2,'(6x,a)') 'nohit called'                  N000180
c                                                                       N000190
      vfarat=-fatn(ifrag)*aptft2/apvft2                                 N000200
      pnrt=(pnrt*float(itrpm)+expon(vfarat))/float(itrpt)               N000210
      pnt(irange)=max(pnt(irange)*pnrt/pn0t,0.0)                        N000220
      pn0t=pnrt                                                         N000230
c                                                                       N000240
c      report return                                                    N000250
      if(merout.gt.0) write(2,'(6x,a)') 'returning from nohit'          N000260
c                                                                       N000270
      return                                                            N000280
c                                                                       N000290
      end                                                               N000300
c                                                                       N000310
c*********************************************************************** N000320
```

```
c*********************************************************************** D000010
c                                                                       D000020
      subroutine dnsty(ifrag,apvft2,irange,itrpm,itrpt,fdrt,fd0t,fdt)   D000030
c                                                                       D000040
c      accumulates fragment density                                     D000050
c                                                                       D000060
c      common blocks                                                    D000070
      common/error/ meror,irerr,iferr,merout                            D000080
      common/repls/ nreps,fnreps                                        D000090
      common/fragd/ famil(900),vfifs(900),farma(900),                   D000100
     +              fatn(900),ezldeg(900)                               D000110
c                                                                       D000120
c      local array                                                      D000130
      dimension fdt(97)                                                 D000140
c                                                                       D000150
c      report call                                                      D000160
      if(merout.gt.0) write(2,'(6x,a)') 'dnsty called'                  D000170
c                                                                       D000180
      fdrt=(fdrt*float(itrpm)+fatn(ifrag)/apvft2)/float(itrpt)          D000190
      fd1t=fdrt/fnreps                                                  D000200
      fdt(irange)=fdt(irange)+fd1t-fd0t                                 D000210
      fd0t=fd1t                                                         D000220
c                                                                       D000230
c      report return                                                    D000240
      if(merout.gt.0) write(2,'(6x,a)') 'returning from dnsty'          D000250
c                                                                       D000260
      return                                                            D000270
c                                                                       D000280
      end                                                               D000290
c                                                                       D000300
c*********************************************************************** D000310
```

```
c*********************************************************************** R000010
c                                                                       R000020
      subroutine ricoc(slcnst,vfrag,aeldeg,selrad,vrico,ricrad)         R000030
c                                                                       R000040
c      determines ricochet conditions                                   R000050
c                                                                       R000060
c      common block                                                     R000070
      common/error/ meror,irerr,iferr,merout                            R000080
c                                                                       R000090
c      report call                                                      R000100
      if(merout.gt.0) write(2,'(2x,a)') 'ricoc called'                  R000110
c                                                                       R000120
      vrat1=-0.01597*aeldeg*aeldeg+0.02156*aeldeg+0.96170               R000130
      vrat2=-0.00861*aeldeg*aeldeg+0.00692*aeldeg+0.96302               R000140
      vrat3=-0.00387*aeldeg*aeldeg-0.00414*aeldeg+0.95592               R000150
      vrat4=-0.00342*aeldeg*aeldeg-0.00097*aeldeg+0.94090               R000160
      vrat5=-0.00243*aeldeg*aeldeg-0.00520*aeldeg+0.93080               R000170
      vrat6=-0.00188*aeldeg*aeldeg-0.00821*aeldeg+0.93802               R000180
c                                                                       R000190
      erat1= 0.13829*aeldeg*aeldeg-0.98645*aeldeg+2.81550               R000200
      erat2= 0.08549*aeldeg*aeldeg-0.78423*aeldeg+2.90120               R000210
      erat3= 0.07515*aeldeg*aeldeg-0.73919*aeldeg+3.10560               R000220
      erat4= 0.02142*aeldeg*aeldeg-0.37397*aeldeg+2.78580               R000230
      erat5= 0.01707*aeldeg*aeldeg-0.32521*aeldeg+2.80920               R000240
      erat6= 0.01369*aeldeg*aeldeg-0.29580*aeldeg+2.82620               R000250
c                                                                       R000260
      if((slcnst.ge.0.25).and.(slcnst.lt.0.5)) then                     R000270
        vrat=vrat1+(slcnst-0.25)/0.25*(vrat2-vrat1)                     R000280
        erat=erat1+(slcnst-0.25)/0.25*(erat2-erat1)                     R000290
        go to 100                                                       R000300
      endif                                                             R000310
c                                                                       R000320
      if((slcnst.ge.0.5).and.(slcnst.lt.1.0)) then                      R000330
        vrat=vrat2+(slcnst-0.5)/0.5*(vrat3-vrat2)                       R000340
        erat=erat2+(slcnst-0.5)/0.5*(erat3-erat2)                       R000350
        go to 100                                                       R000360
      endif                                                             R000370
c                                                                       R000380
      if((slcnst.ge.1.0).and.(slcnst.lt.2.0)) then                      R000390
        vrat=vrat3+(slcnst-1.0)*(vrat4-vrat3)                           R000400
        erat=erat3+(slcnst-1.0)*(erat4-erat3)                           R000410
        go to 100                                                       R000420
      endif                                                             R000430
c                                                                       R000440
      if((slcnst.ge.2.0).and.(slcnst.lt.3.0)) then                      R000450
        vrat=vrat4+(slcnst-2.0)*(vrat5-vrat4)                           R000460
        erat=erat4+(slcnst-2.0)*(erat5-erat4)                           R000470
        go to 100                                                       R000480
      endif                                                             R000490
c                                                                       R000500
      vrat=vrat5+(slcnst-3.0)*(vrat6-vrat5)                             R000510
      erat=erat5+(slcnst-3.0)*(erat6-erat5)                             R000520
c                                                                       R000530
  100 vrico=vrat*vfrag                                                  R000540
      ricrad=-selrad*erat                                               R000550
c                                                                       R000560
c      report return                                                    R000570
      if(merout.gt.0) write(2,'(2x,a)') 'returning from ricoc'          R000580
c                                                                       R000590
      return                                                            R000600
c                                                                       R000610
      end                                                               R000620
c                                                                       R000630
c*********************************************************************** R000640
```

```
c*********************************************************************** R000010
c                                                                       R000020
      subroutine reini(irico,vrico,ricrad,wspeed,wdrdeg, vfrag,selrad,  R000030
     +                 seldeg,aelrad,aeldeg,deldst,mfrht1)              R000040
c                                                                       R000050
c     reinitializes the velocity integration variables after ricochet   R000060
c                                                                       R000070
c      common blocks                                                    R000080
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   R000090
      common/error/ meror,irerr,iferr,merout                            R000100
      common/rkvar/ rkv(7)                                              R000110
c                                                                       R000120
c      report call                                                      R000130
      if(merout.gt.0) write(2,'(2x,a)') 'reini called'                  R000140
c                                                                       R000150
c      increment the ricochet count                                     R000160
      irico=irico+1                                                     R000170
c                                                                       R000180
c      assign the ricochet values to the fragment variables             R000190
      vfrag=vrico                                                       R000200
      selrad=ricrad                                                     R000210
      seldeg=raddeg*selrad                                              R000220
      aelrad=abs(selrad)                                                R000230
      aeldeg=abs(seldeg)                                                R000240
c                                                                       R000250
c     assign the fragment variable values to the Runge-Kutta variables  R000260
      rkv(4)=vfrag*sin(selrad)                                          R000270
      if((wdrdeg.eq.0.0).or.(wspeed.eq.0.0)) then                       R000280
        rkv(2)=vfrag*cos(selrad)                                        R000290
        rkv(6)=0.0                                                      R000300
      else                                                              R000310
        angrad=atan(rkv(2)/rkv(6))                                      R000320
        rkv(2)=vfrag*cos(selrad)*sin(angrad)                            R000330
        rkv(6)=vfrag*cos(selrad)*cos(angrad)                            R000340
      endif                                                             R000350
c                                                                       R000360
      call dstep(rkv(5),seldeg,selrad, deldst)                          R000370
c                                                                       R000380
      mfrht1=0                                                          R000390
c                                                                       R000400
c      report return                                                    R000410
      if(merout.gt.0) write(2,'(2x,a)') 'returning from reini'          R000420
c                                                                       R000430
      return                                                            R000440
c                                                                       R000450
      end                                                               R000460
c                                                                       R000470
c*********************************************************************** R000480
```

```
c*********************************************************************** H000010
c                                                                       H000020
      subroutine hpacc                                                  H000030
c                                                                       H000040
c      accumulates hit probabilities                                    H000050
c                                                                       H000060
c      common blocks                                                    H000070
      common/error/ meror,irerr,iferr,merout                            H000080
      common/repls/ nreps,fnreps                                        H000090
      common/rangs/ nrsegs,nrsegp                                       H000100
      common/phitt/ ph(97),phd(97),phb(97),phm(97)                      H000110
      common/pmiss/ pn(97),pnd(97),pnb(97),pnm(97)                      H000120
c                                                                       H000130
c      report call                                                      H000140
      if(merout.gt.0) write(2,'(2x,a)') 'hpacc called'                  H000150
c                                                                       H000160
      do 100 irange=1,nrsegs                                            H000170
        ph (irange)=ph (irange)+(1.0-pn (irange))/fnreps                H000180
        phd(irange)=phd(irange)+(1.0-pnd(irange))/fnreps                H000190
        phb(irange)=phb(irange)+(1.0-pnb(irange))/fnreps                H000200
        phm(irange)=phm(irange)+(1.0-pnm(irange))/fnreps                H000210
  100 continue                                                          H000220
c                                                                       H000230
c      report return                                                    H000240
      if(merout.gt.0) write(2,'(2x,a)') 'returning from hpacc'          H000250
c                                                                       H000260
      return                                                            H000270
c                                                                       H000280
      end                                                               H000290
c                                                                       H000300
c*********************************************************************** H000310

c*********************************************************************** O000010
c                                                                       O000020
      subroutine outpt                                                  O000030
c                                                                       O000040
c      writes output to appropriate files                               O000050
c                                                                       O000060
c      common blocks                                                    O000070
      common/phitt/ ph(97),phd(97),phb(97),phm(97)                      O000080
      common/denst/ fd(97),fdd(97),fdb(97),fdm(97)                      O000090
c                                                                       O000100
c      output the hit probabilities                                     O000110
      call outfi('fp.php',ph,phd,phb,phm)                               O000120
c                                                                       O000130
c      output the fragment densities                                    O000140
      call outfi('fp.pfd',fd,fdd,fdb,fdm)                               O000150
c                                                                       O000160
      return                                                            O000170
c                                                                       O000180
      end                                                               O000190
c                                                                       O000200
c*********************************************************************** O000210
```

```
c*********************************************************************** O000010
c                                                                       O000020
      subroutine outfi(fname,v,vd,vb,vm)                                O000030
c                                                                       O000040
c      outputs hit probabilities or fragment densities                  O000050
c                                                                       O000060
c      common blocks                                                    O000070
      common/repls/ nreps,fnreps                                        O000080
      common/donor/ nunits,htstft,htbaft                                O000090
      common/rangs/ nrsegs,nrsegp                                       O000100
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin           O000110
      common/arang/ rangem(97),ahfrn(97),ahtop(97)                      O000120
      common/names/ dname,aname                                         O000130
      common/acptr/ htacft,wdacft,dpacft,htacfm                         O000140
c                                                                       O000150
c      local arrays                                                     O000160
      dimension v(97),vd(97),vb(97),vm(97)                              O000170
c                                                                       O000180
c      type declarations                                                O000190
      character fname*6,dname*40,aname*40                               O000200
c                                                                       O000210
      open(3,file=fname,status='unknown')                               O000220
c                                                                       O000230
      write(3,'(3a)') char(39),dname,char(39)                           O000240
      write(3,'(2i3,1p2e16.8)') nreps,nunits,htstft,htbaft              O000250
      write(3,'(3a)') char(39),aname,char(39)                           O000260
      write(3,'(1p3e16.8)') htacft,wdacft,dpacft                        O000270
      write(3,'(1p2e16.8)') rngmin,rngmax                               O000280
      do 100 irange=1,nrsegs                                            O000290
        write(3,'(1p5e16.8)') rangem(irange),vd(irange),                O000300
     +                         vb(irange),vm(irange),v(irange)          O000310
  100 continue                                                          O000320
c                                                                       O000330
      close(3)                                                          O000340
c                                                                       O000350
      return                                                            O000360
c                                                                       O000370
      end                                                               O000380
c                                                                       O000390
c*********************************************************************** O000400
```

```
c*********************************************************************** R000010
c                                                                       R000020
      function rnval(valmin,valmax,rndnxt)                              R000030
c                                                                       R000040
c      generates a random value between two specified values            R000050
c                                                                       R000060
c      type declaration                                                 R000070
      double precision rndnxt                                           R000080
c                                                                       R000090
      rnval=valmin+(valmax-valmin)*rndom(rndnxt)                        R000100
c                                                                       R000110
      return                                                            R000120
c                                                                       R000130
      end                                                               R000140
c                                                                       R000150
c*********************************************************************** R000160

c*********************************************************************** R000010
c                                                                       R000020
      function rndom(uix)                                               R000030
c                                                                       R000040
c      generates random numbers                                         R000050
c                                                                       R000060
c      type declarations                                                R000070
      double precision ua,up,uix,uiy,ub15,ub16,                         R000080
     +                 uxhi,uxalo,ulftlo,ufhi,uk                        R000090
c                                                                       R000100
c      data statement                                                   R000110
      data ua/16807.0d0/,ub15/32768.0d0/,                               R000120
     +     ub16/65536.0d0/,up/2147483647.0d0/                           R000130
c                                                                       R000140
      if(uix.eq.0) uix=uiy                                              R000150
  100 uxhi=uix/ub16                                                     R000160
      uxhi=uxhi-dmod(uxhi,1.0d0)                                        R000170
      uxalo=(uix-uxhi*ub16)*ua                                          R000180
      ulftlo=uxalo/ub16                                                 R000190
      ulftlo=ulftlo-dmod(ulftlo,1.0d0)                                  R000200
      ufhi=uxhi*ua+ulftlo                                               R000210
      uk=ufhi/ub15                                                      R000220
      uk=uk-dmod(uk,1.0d0)                                              R000230
      uix=(((uxalo-ulftlo*ub16)-up)+(ufhi-uk*ub15)*ub16)+uk             R000240
      if(uix.lt.0.0) uix=uix+up                                         R000250
      uiy=uix                                                           R000260
      rndom=int((uix*4.656612875d-10)*1.0d06)/1.0d06                    R000270
      if(rndom.eq.0.0) go to 100                                        R000280
c                                                                       R000290
      return                                                            R000300
c                                                                       R000310
      end                                                               R000320
c                                                                       R000330
c*********************************************************************** R000340
```

```
c*********************************************************************** V000010
c                                                                       V000020
      function vcrjr(abjr,cjr,hmm,fdmm)                                 V000030
c                                                                       V000040
c      computes the Jacobs-Roslund critical velocity                    V000050
c                                                                       V000060
c      common block                                                     V000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   V000080
c                                                                       V000090
      if(fdmm.eq.0.0) then                                              V000100
        vcrjr=bignum                                                    V000110
      else                                                              V000120
        vcrjr=abjr*(1.0+cjr*hmm/fdmm)/sqrt(fdmm)                        V000130
      endif                                                             V000140
c                                                                       V000150
      return                                                            V000160
c                                                                       V000170
      end                                                               V000180
c                                                                       V000190
c*********************************************************************** V000200

c*********************************************************************** V000010
c                                                                       V000020
      function vcrth(tc,ath,bth,hcm,facm2,fmgm)                         V000030
c                                                                       V000040
c      computes the THOR critical velocity                              V000050
c                                                                       V000060
c      common block                                                     V000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   V000080
c                                                                       V000090
      if(fmgm.eq.0.0) then                                              V000100
        vcrth=bignum                                                    V000110
      else                                                              V000120
        vcrth=tc*((hcm*facm2)**ath)*(fmgm**bth)                         V000130
      endif                                                             V000140
c                                                                       V000150
      return                                                            V000160
c                                                                       V000170
      end                                                               V000180
c                                                                       V000190
c*********************************************************************** V000200

c*********************************************************************** E000010
c                                                                       E000020
      function expon(arg)                                               E000030
c                                                                       E000040
c      computes the exponential function                                E000050
c      limited for large negative arguments                             E000060
c                                                                       E000070
c      common block                                                     E000080
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum   E000090
c                                                                       E000100
      expon=max(exp(arg),smlnum)                                        E000110
c                                                                       E000120
      return                                                            E000130
c                                                                       E000140
      end                                                               E000150
c                                                                       E000160
c*********************************************************************** E000170
```

APPENDIX C:

LETHALITY AND VULNERABILITY DATA

INTENTIONALLY LEFT BLANK.

The data required for M107 and TOW-2A lethality and vulnerability descriptions is summarized in Tables C-1 through C-5.

Table C-1. Weapon Dimensions and Materials

Component	M107	TOW-2A
Weapon:		
Outside Diameter	6.10 in	6.187 in
Length	23.90 in	60.000 in*
Warhead:		
Outside Diameter	6.10 in	5.850 in
Length	23.90 in	25.570 in
Casing Material	steel	aluminum
Casing Thickness	0.63 in	0.050 in
Explosive Type	Comp. B	LX-14
Charge Diameter	4.84 in	5.750 in
Charge Length	20.30 in	9.574 in
Rocket Motor:		
Outside Diameter	N/A	5.837 in
Length	N/A	18.700 in
Casing Material	N/A	steel
Casing Thickness	N/A	0.055 in
Propellant Type	N/A	GCV
Charge Diameter	N/A	5.782 in
Charge Length	N/A	6.320 in
Container:		
Material	N/A	steel
Outside Diameter	N/A	10.000 in
Thickness	N/A	0.040 in
Pallet:		
Orientation	vertical	horizontal
Arrangement (H × W × D)	1 × 4 × 2	3 × 1 × 4
Weapon Spacing	1.20 in	1.000 in*
Elevation	8.10 in	8.100 in*
Height (with pallet)	32.00 in	40.220 in
Width	28.00 in	60.000 in
Depth	13.40 in	43.160 in

* estimated value

Table C-2. Energetic Material Performance Parameters

Material	Density (g/cm^2)	Detonation Pressure (kbar)	Gurney Constant (m/s)	Product Gases (mole/g)	Molecular Weight	Heat of Detonation (cal/g)
LX-14	1.83	370.0	2,948.0	Not Required	Not Required	Not Required
GCV Propellant	1.8	Unavailable	Unavailable	0.042156	24.635	890.0

Table C-3. Jacobs-Roslund Constants

Material	a_{jr} $\left(mm^{3/2}/\mu s\right)$	b_{jr}	c_{jr}
Composition B	3.065	0.0	1.70
LX-14*	3.6	0.0	1.59
GCV Propellant*	4.8	0.0	1.3

* estimated values

Table C-4. THOR Velocity Equation Constants

Material	a_v	b_v	c_v	d_v	e_v
mild steel	3.6901	0.889	–0.945	0.019	1.262
aluminum	3.9356	1.029	–1.072	–0.139	1.251

Table C-5. THOR Mass Equation Constants

Material	a_m	b_m	c_m	d_m
mild steel	–2.478	0.138	0.835	0.761

APPENDIX D:

DONOR FRAGMENTATION GENERATOR LISTING

INTENTIONALLY LEFT BLANK.

```
c*********************************************************************** D000010
c                                                                       D000020
      program dogen                                                     D000030
c                                                                       D000040
c                    DONOR FRAGMENTATION DATA GENERATOR                 D000050
c                           (Horizontal Storage)                        D000060
c                                                                       D000070
c      common blocks                                                    D000080
      common/frgwh/ iazwh(2500),fmwhgm(2500)                            D000090
      common/frgrm/ iazrm(2500),fmrmgm(2500)                            D000100
c                                                                       D000110
c      type declaration                                                 D000120
      double precision rndnxt                                           D000130
c                                                                       D000140
c      define constants                                                 D000150
      call const                                                        D000160
c                                                                       D000170
c      read and compute warhead parameters                              D000180
      call elein('wh.inp',nfrwh,fmavwh,vfwhfs,plzwh,fmnwh)              D000190
c                                                                       D000200
c      read and compute rocket motor parameters                         D000210
      call elein('rm.inp',nfrrm,fmavrm,vfrmfs,plzrm,fmnrm)              D000220
c                                                                       D000230
c      input the randomization seed                                     D000240
      call rndin(rndnxt)                                                D000250
c                                                                       D000260
c      generate the warhead fragment distriubtion                       D000270
      call frdst(rndnxt,nfrwh,fmavwh,fmnwh,nazwh,lfrwh,iazwh,fmwhgm,    D000280
     +           iazwhx)                                                D000290
c                                                                       D000300
c      generate the rocket motor fragment distriubtion                  D000310
      call frdst(rndnxt,nfrrm,fmavrm,fmnrm,nazrm,lfrrm,iazrm,fmrmgm,    D000320
     +           iazrmx)                                                D000330
c                                                                       D000340
c      generate fragmentation data file                                 D000350
      call mkfil(plzwh,nazwh,lfrwh,nfrwh,vfwhfs,fmnwh,iazwhx,           D000360
     +           plzrm,nazrm,lfrrm,nfrrm,vfrmfs,fmnrm,iazrmx)           D000370
c                                                                       D000380
      stop ' '                                                          D000390
c                                                                       D000400
      end                                                               D000410
c                                                                       D000420
c*********************************************************************** D000430

c*********************************************************************** C000010
c                                                                       C000020
      subroutine const                                                  C000030
c                                                                       C000040
c      defines constants                                                C000050
c                                                                       C000060
c      common block                                                     C000070
      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg                        C000080
c                                                                       C000090
      pi=4.0*atan(1.0)                                                  C000100
      twth=2.0/3.0                                                      C000110
      dazdeg=10.0                                                       C000120
      idzdeg=int(dazdeg)                                                C000130
      azxdeg=110.0                                                      C000140
c                                                                       C000150
      return                                                            C000160
c                                                                       C000170
      end                                                               C000180
c                                                                       C000190
c*********************************************************************** C000200
```

```
c*********************************************************************** E000010
c                                                                       E000020
      subroutine elein(infile,nfrag,frmav,vffs,plzdeg,frmmin)           E000030
c                                                                       E000040
c      reads and computes cylindrical element parameters                E000050
c                                                                       E000060
c      common block                                                     E000070
      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg                        E000080
c                                                                       E000090
c      type declaration                                                 E000100
      character*6 infile                                                E000110
c                                                                       E000120
c      open data file                                                   E000130
      open(3,file=infile,status='old')                                  E000140
c                                                                       E000150
c      read casing data                                                 E000160
      read(3,*) rhca,romm,drmm,htmm,plzdeg,frmmin                       E000170
c                                                                       E000180
c      read fill data                                                   E000190
      read(3,*) rhfl,pdfl,vgfl                                          E000200
      if(pdfl.eq.0.0) read(3,*) fln,flm,flq                             E000210
c                                                                       E000220
c      close data file                                                  E000230
      close(3)                                                          E000240
c                                                                       E000250
c      convert dimensions to centimeters                                E000260
      rocm=0.1*romm                                                     E000270
      drcm=0.1*drmm                                                     E000280
      htcm=0.1*htmm                                                     E000290
c                                                                       E000300
c      compute the casing thickness and inside diameter in inches       E000310
      drin=drmm/25.4                                                    E000320
      diin=(romm-drmm)/25.4                                             E000330
c                                                                       E000340
c      compute the inside radius in centimeters                         E000350
      ricm=rocm-drcm                                                    E000360
      ricm2=ricm*ricm                                                   E000370
c                                                                       E000380
c      compute the casing mass in grams                                 E000390
      camgm=rhca*pi*(rocm*rocm-ricm2)*htcm                              E000400
      write(*,'(a,1pe12.5,a)') '            casing mass =',camgm,' g'   E000410
c                                                                       E000420
c      compute the fill mass in grams                                   E000430
      flmgm=rhfl*pi*ricm2*htcm                                          E000440
      write(*,'(a,1pe12.5,a)') '             fill mass =',flmgm,' g'    E000450
c                                                                       E000460
c      compute the ratio of casing to fill mass                         E000470
      camfl=camgm/flmgm                                                 E000480
c                                                                       E000490
c      compute the detonation pressure if necessary                     E000500
      if(pdfl.eq.0.0) pdfl=15.58*rhfl*rhfl*fln*sqrt(flm*flq)            E000510
      write(*,'(a,1pe12.5,a)') '   detonation pressure =',pdfl,' kbar'   E000520
c                                                                       E000530
c      compute the Gurney constant if necessary                         E000540
      if(vgfl.eq.0.0) vgfl=233.0*sqrt(pdfl)/rhfl**0.6                   E000550
      write(*,'(a,1pe12.5,a)') '       Gurney constant =',vgfl,' m/s'   E000560
c                                                                       E000570
c      compute the fragment velocity                                    E000580
      vfms=vgfl/sqrt(camfl+0.5)                                         E000590
      vffs=3.2808399*vfms                                               E000600
      write(*,'(a,1pe12.5,a)') '    fragment velocity =',vfms,' m/s'    E000610
c                                                                       E000620
c      compute the average fragment mass in grams                       E000630
      frmav=                                                            E000640
     + 6.762e2*sqrt(1.0+0.5*camfl)*(drin*((diin+drin)**1.5)/diin)/pdfl  E000650
      write(*,'(a,1pe12.5,a)') ' average fragment mass =',frmav,' g'    E000660
c                                                                       E000670
c      compute the number of fragments (adjust sector size if needed)   E000680
  100 nfrag=int(camgm*azxdeg/(360.0*frmav))                             E000690
      if(nfrag.gt.2500) then                                            E000700
        azxdeg=2500.0*azxdeg/float(nfrag)                               E000710
        go to 100                                                       E000720
      endif                                                             E000730
      write(*,'(a,i5)')              '  number of fragments =',nfrag    E000740
      write(*,'(a,1pe12.5,a/)') ' azimuthal sector size =',azxdeg,' deg' E000750
c                                                                       E000760
      return                                                            E000770
c                                                                       E000780
      end                                                               E000790
c                                                                       E000800
c*********************************************************************** E000810
```

```
c*********************************************************************** F000010
c                                                                       F000020
      subroutine frdst(rndnxt,nfrag,frmav,frmmin,naz,lfrag,iaz,frmgm,   F000030
     +                 iazmax)                                          F000040
c                                                                       F000050
c      generates a fragment distriubtion                                F000060
c                                                                       F000070
c      common block                                                     F000080
      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg                        F000090
c                                                                       F000100
c      type declaration                                                 F000110
      double precision rndnxt                                           F000120
c                                                                       F000130
c      local arrays                                                     F000140
      dimension iaz(2500),frmgm(2500)                                   F000150
c                                                                       F000160
c      generate fragment masses with Mott distribution                  F000170
c      and azimutal angles with even distribution                       F000180
      hfrmav=0.5*frmav                                                  F000190
      iazmax=0                                                          F000200
      lfrag=0                                                           F000210
      do 100 ifrag=1,nfrag                                              F000220
        rnddum=rndom(rndnxt)                                            F000230
        frmgm(ifrag)=-hfrmav*alog(rnddum*rnddum)                        F000240
        iaz(ifrag)=idzdeg*int(azxdeg*rndom(rndnxt)/dazdeg)+idzdeg       F000250
        iazmax=max(iazmax,iaz(ifrag))                                   F000260
        if(frmgm(ifrag).gt.frmmin) lfrag=lfrag+1                        F000270
  100 continue                                                          F000280
      naz=iazmax/idzdeg                                                 F000290
c                                                                       F000300
      return                                                            F000310
c                                                                       F000320
      end                                                               F000330
c                                                                       F000340
c*********************************************************************** F000350
```

```
c*********************************************************************** M000010
c                                                                       M000020
      subroutine mkfil(plzwh,nazwh,lfrwh,nfrwh,vfwhfs,fmnwh,iazwhx,     M000030
     +                 plzrm,nazrm,lfrrm,nfrrm,vfrmfs,fmnrm,iazrmx)     M000040
c                                                                       M000050
c      generates fragmentation data file                                M000060
c                                                                       M000070
c      common blocks                                                    M000080
      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg                        M000090
      common/frgwh/ iazwh(2500),fmwhgm(2500)                            M000100
      common/frgrm/ iazrm(2500),fmrmgm(2500)                            M000110
c                                                                       M000120
c      output fragmentation data with                                   M000130
c      azimuthal and polar angles interchanged                          M000140
      naz=max(nazwh,nazrm)                                              M000150
      iazmax=max(iazwhx,iazrmx)                                         M000160
      lfrag=lfrwh+lfrrm                                                 M000170
      open(3,file='fr.out',status='unknown')                            M000180
      write(3,'(3x,1p2e15.7)') dazdeg,max(plzwh,plzrm)                  M000190
      write(3,'(3x,3i5)') naz,lfrag,1                                   M000200
      write(4,1pe15.7) 0.04                                             M000210
      write(4,1p4e15.7) 3.6901,0.889,-0.945,0.019                       M000220
      write(4,1p4e15.7) -2.478,0.138,0.835,0.761                        M000230
      write(3,'(3x,1p4e15.7)') (0.05,iz=1,naz)                          M000240
      do 100 iwrite=idzdeg,iazmax,idzdeg                                M000250
        call adfrg(iwrite,nfrwh,fmwhgm,iazwh,fmnwh,vfwhfs)              M000260
        call adfrg(iwrite,nfrrm,fmrmgm,iazrm,fmnrm,vfrmfs)              M000270
  100 continue                                                          M000280
      close(3)                                                          M000290
c                                                                       M000300
      return                                                            M000310
c                                                                       M000320
      end                                                               M000330
c                                                                       M000340
c*********************************************************************** M000350

c*********************************************************************** A000010
c                                                                       A000020
      subroutine adfrg(iwrite,nfrag,frmgm,iaz,fmn,vffs)                 A000030
c                                                                       A000040
c                                                                       A000050
c      common blocks                                                    A000060
      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg                        A000070
c                                                                       A000080
c      local arrays                                                     A000090
      dimension iaz(2500),frmgm(2500)                                   A000100
c                                                                       A000110
      do 100 ifrag=1,nfrag                                              A000120
        if(frmgm(ifrag).gt.fmn) then                                    A000130
          if(iaz(ifrag).eq.iwrite) then                                 A000140
            acm2=0.5199*frmgm(ifrag)**twth                              A000150
            ain2=0.1550003*acm2                                         A000160
            frmlb=2.2046226e-3*frmgm(ifrag)                             A000170
            frmgr=7000.0*frmlb                                          A000180
            aminlb=ain2/frmlb                                           A000190
            write(3,'(i3,1p4e15.7)') iaz(ifrag),frmgr,aminlb,vffs,1.5   A000200
          endif                                                         A000210
        endif                                                           A000220
  100 continue                                                          A000230
c                                                                       A000240
      return                                                            A000250
c                                                                       A000260
      end                                                               A000270
c                                                                       A000280
c*********************************************************************** A000290
```

```
c*********************************************************************** R000010
c                                                                       R000020
      subroutine rndin(rndnxt)                                          R000030
c                                                                       R000040
c      obtains randomization seed input                                 R000050
c                                                                       R000060
c      type declarations                                                R000070
      double precision rndinp,rndnxt                                    R000080
c                                                                       R000090
  100 write(*,'(a\)') ' Enter Monte Carlo Seed (1 to 2147483646): '     R000100
      read(*,*,err=100) rndinp                                          R000110
      if(rndinp.lt.1.0.or.rndinp.gt.2147483646.0) go to 100             R000120
c                                                                       R000130
      rnddum=rndom(rndinp)                                              R000140
      rndnxt=0.0d0                                                      R000150
c                                                                       R000160
      return                                                            R000170
c                                                                       R000180
      end                                                               R000190
c                                                                       R000200
c*********************************************************************** R000210

c*********************************************************************** R000010
c                                                                       R000020
      function rndom(uix)                                               R000030
c                                                                       R000040
c      generates random numbers                                         R000050
c                                                                       R000060
c      type declarations                                                R000070
      double precision ua,up,uix,uiy,ub15,ub16,uxhi,uxalo,ulftlo,ufhi,  R000080
     +                 uk                                             R000090
c                                                                       R000100
c      data statement                                                   R000110
      data ua/16807.0d0/,ub15/32768.0d0/,ub16/65536.0d0/,               R000120
     +     up/2147483647.0d0/                                           R000130
c                                                                       R000140
      if(uix.eq.0) uix=uiy                                              R000150
  100 uxhi=uix/ub16                                                     R000160
      uxhi=uxhi-dmod(uxhi,1.0d0)                                        R000170
      uxalo=(uix-uxhi*ub16)*ua                                          R000180
      ulftlo=uxalo/ub16                                                 R000190
      ulftlo=ulftlo-dmod(ulftlo,1.0d0)                                  R000200
      ufhi=uxhi*ua+ulftlo                                               R000210
      uk=ufhi/ub15                                                      R000220
      uk=uk-dmod(uk,1.0d0)                                              R000230
      uix=(((uxalo-ulftlo*ub16)-up)+(ufhi-uk*ub15)*ub16)+uk             R000240
      if(uix.lt.0.0) uix=uix+up                                         R000250
      uiy=uix                                                           R000260
      rndom=int((uix*4.656612875d-10)*1.0d06)/1.0d06                    R000270
      if(rndom.eq.0.0) go to 100                                        R000280
c                                                                       R000290
      return                                                            R000300
c                                                                       R000310
      end                                                               R000320
c                                                                       R000330
c*********************************************************************** R000340
```

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
2	ADMINISTRATOR DEFENSE TECHNICAL INFO CTR ATTN DTIC DDA CAMERON STATION ALEXANDRIA VA 22304-6145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TA 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TL 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TP 2800 POWDER MILL RD ADELPHI MD 20783-1145
	ABERDEEN PROVING GROUND
5	DIR USARL ATTN AMSRL OP AP L (305)

NO. OF COPIES ORGANIZATION

1 DIRECTOR
USA TECH CTR FOR EXPLOSIVE SAFETY
ATTN SMCAC ESL C DOYLE
SAVANNA IL 61074-9639

1 DIRECTOR
USA TECH CTR FOR EXPLOSIVE SAFETY
ATTN SMCAC DO N MCCULLOUGH
SAVANNA IL 61074-9639

3 US ARMY WATERWAYS EXP STATION
ATTN CEWES SE
K DAVIS
M FORD
C JOACHIM
3909 HALLS FERRY RD
VICKSBURG MS 39180-6199

2 CHAIRMAN
DOD EXPLOSIVE SAFETY BOARD
ATTN DDESB KT
J WARD
C CANADA
2461 EISENHOWER AVE
ALEXANDRIA VA 22331-0600

4 DEFENSE AMMUNITION LOGISTICS ACTIVITY
ATTN AMCPM AL
R ROSSI
R HO
E GOON
B WILLIAMSON
PICATINNY ARSENAL NJ 07806-5000

1 COMMANDER
ATTN SMCAR AEE W
P LU
US ARMY ARDEC
PICATINNY ARSENAL NJ 07806-5000

1 COMMANDER
ATTN SMCAR AEM
B WILLIAMSON
US ARMY ARDEC
PICATINNY ARSENAL NJ 07806-5000

1 COMMANDER
ATTN SMCAR ASA
D MILLER
US ARMY ARDEC
PICATINNY ARSENAL NJ 07806-5000

NO. OF COPIES ORGANIZATION

1 OFFICER IN CHARGE
WHITE OAK LABORATORY NSWC
ATTN R15 M SWISDAK
10901 NEW HAMPSHIRE AVE
SILVER SPRING MD 20902-5000

2 NAVAL FACILITIES ENGINEERING SERV CTR
ATTN CODE LS1
J TANCRETO
R MURTHE
PORT HUENEME CA 93043

1 DIRECTOR
EXPLOSIVES HAZARD REDUCTION DIR
ATTN ASC YOCO EHR
J JENUS
EGLIN AFB FL 32542

2 US ARMY CORPS OF ENGINEERS
ATTN CEHND ED CS
R WRIGHT
R HASSE
PO BOX 1600
HUNTSVILLE AL 35807-4301

ABERDEEN PROVING GROUND, MD

13 DIR, USARL
ATTN: AMSRL-WT-T, W. MORRISON
AMSRL-WT-TB,
P. BAKER
R. FREY
O. LYMAN
J. WATSON
F. GREGORY
V. BOYLE
W. HILLSTROM
E. MCDOUGAL
W. LAWRENCE
K. BENJAMIN
T. DORSEY
AMSRL-SL-BV, J. PLOSONKA

2 DIR, USA AMSAA
ATTN: AMXSY-GC,
E. CHRISTMAN
A. WONG

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-TR-949 Date of Report January 1996

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001,APG,MD

POSTAGE WILL BE PAID BY ADDRESSEE

DIRECTOR
U.S. ARMY RESEARCH LABORATORY
ATTN: AMSRL-WT-TB
ABERDEEN PROVING GROUND, MD 21005-5066